UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Statistical Regular Pavings and their Applications

A thesis submitted in partial fulfilment of the requirements for the Degree of
Doctor of Philosophy in Statistics

by Gloria Teng Ai Hui

under the supervision of
Dr. Raazesh Sainudiin
and
Dr. Dominic Lee

Department of Mathematics and Statistics
University of Canterbury

January 18, 2013

[This page intentionally left blank]

Acknowledgements

This research project has opened up frontiers that I would never thought I was capable of delving into, and has given me a set of skills that I never thought I could acquire, and for this, I am wholly indebted to my senior supervisor, Dr. Raazesh Sainuddin. I thank him for being a willing and supportive supervisor, teacher, friend, and fatherly figure. I have learnt much, and am still learning from him.

I would also like to thank my assistant supervisor, Dr. Dominic Lee, for recommending me to work under Dr. Sainuddin's supervision, and for always being the voice of consciousness, keeping me in check.

Jennifer Harlow is the main programmer for most of the algorithms found in this project. She has been a critical voice of reasoning when I find myself going in circles, and has always been there to give a helping hand in the programming aspects of this project.

Special thanks also to Brendan Bycroft for writing the automated make tools for the `C++` library that we have developed, and also to Dr. Sandro Mattarei for kindly sharing the proof of the asypmtotic sum of partial Catalan numbers.

Thank you to Dr .Kenneth Kuhn, formerly from the Civil Engineering Department at the University of Canterbury, for being willing to work with us and generously sharing his experience and data for part of this research.

A big shout of thanks to the staff at the department, especially Steve Gourdie, Allen Witt, and Paul Brouwers for assisting me with all computer-related issues. Thanks too, to the wonderful department academics and staff who have never failed to make my day with friendly smiles and words of encouragement.

To my postgraduate colleagues, what would postgraduate life be without all of you, especially during the earthquake and post-earthquake times. We supported each other and were more united than before, determined to continue to create a stimulating environment for research and study despite the lack of resources when our offices and common social places were all closed due to the earthquake. These are memories that will never be forgotten.

Finally, to my dearest family, to my friends, and to the One Above, thank you all for your unconditional love and support, for being with me during the dark times and celebrating good times together. You are all precious to me.

[This page intentionally left blank]

Abstract

We propose using statistical regular pavings (SRPs) as an efficient and adaptive statistical data structure for processing massive, multi-dimensional data. A regular paving (RP) is an ordered binary tree that recursively bisects a box in $\mathbb{R}^d$ along the first widest side. An SRP is extended from an RP by allowing mutable caches of recursively computable statistics of the data. In this study we use SRPs for two major applications: estimating histogram densities and summarising large spatio-temporal datasets. The SRP histograms produced are L_1-consistent density estimators driven by a randomised priority queue that adaptively grows the SRP tree, and formalised as a Markov chain over the space of SRPs. A way to select an estimate is to run a Markov chain over the space of SRP trees, also initialised by the randomised priority queue, but here the SRP tree either shrinks or grows adaptively through pruning or splitting operations. The stationary distribution of the Markov chain is then the posterior distribution over the space of all possible histograms. We then take advantage of the recursive nature of SRPs to make computationally efficient arithmetic averages, and take the average of the states sampled from the stationary distribution to obtain the posterior mean histogram estimate.

We also show that SRPs are capable of summarizing large datasets by working with a dataset containing high frequency aircraft position information. Recursively computable statistics can be stored for variable-sized regions of airspace. The regions themselves can be created automatically to reflect the varying density of aircraft observations, dedicating more computational resources and providing more detailed information in areas with more air traffic. In particular, SRPs are able to very quickly aggregate or separate data with different characteristics so that data describing individual aircraft or collected using different technologies (reflecting different levels of precision) can be stored separately and yet also very quickly combined using standard arithmetic operations.

[This page intentionally left blank]

Contents

Chapter 1

Introduction

1.1 The Data Deluge

"There's simply too much out there, and it's too hard to understand."
The Director of the National Security Agency (King, 2001, p. 1)

This was a statement made in response to the "sheer volume and variety of today's communication" (King, 2001), a result of breakthroughs in data transfer afforded by fibre optics technology. That was a decade ago. In February 2011, Hilbert and López (2011) estimated that current technology is able to "store 2.9×10^{20} optimally compressed bytes, communicate almost 2×10^{21}, and carry out 6.4×10^{18} instructions per second on general-purpose computers." Advances in technology, computing power and storage capabilities have indeed made data easily and increasingly available, resulting in a data deluge, and creating the need to deal with massive or large datasets in an efficient and meaningful manner. Some examples from the business world include tick-by-tick financial data, supermarket sales, credit card transactions, and billing records. Typical examples from sciences include astronomical data, weather data, health care data, brain imaging, and gene expression profiles. In technology, social media data and IP logs are large datasets with a wealth of information to be explored. In reality, the term "massive" or "large" datasets not only implies large amounts of observations, in addition it also comprises the possibly large number of variables associated with each observation, with possible interaction between variables, further increasing the complexity and difficulty for data analyses (Donoho, 2000). For instance, IP logs are associated with web browsing information, financial transactions involved various consumer details, and weather data included geographical information. Kettenring (2009) summarized various view points on the notion of massiveness as follows: "It is the combination of size and complexity of the dataset, resulting in aggravation or even impasse when one attempts to analyse it, that gives rise to the label *massive*."

Given limitations in machine memory and speed, computer-aided statistical decisions based on standard methods and theory that cater primarily to modest datasets, will often be impractical, inappropriate or ineffective for such massive datasets, especially in the presence of updates. In the February 2011 issue of Science Magazine, various issues on the challenges faced when dealing with large datasets were explored. This included a "storage gap" where more data are collected than can be physically stored; and the challenge of accessing, managing, organizing, and extracting meaningful data for efficient usage. Results from a poll conducted by Science Magazine indicated that most researchers require extra help in massive data analysis, emphasising the need for adaptation of existing methods and development of new tools for analysing such large data sets. Lambert (2003) proposed that statistical models developed to handle large data sets should be nonparametric in nature and statistically sufficient, as well as having the ability to be dynamic and easily initialised. Kettenring (2009) suggested guided data visualization techniques to view the data from different perspectives as well as data aggregation techniques to reduce the need to store data. Both Lambert (2003) and Kettenring (2009) emphasized the need for inter-disciplinary collaboration where experts from various fields would contribute to the analyses within their expertise.

Ongoing research includes developing algorithms for various tasks such as classification, range query, network learning, information retrieval, etc. Friedman and Getoor (1999) proposed an algorithm using sufficient statistics based on constraints to avoid unnecessarily collecting statistics, and thus speed up the process of learning Bayesian networks. AD-trees were used by Moore and Soon Lee (1998), which allows for splits on any dimensions at all nodes. Only sufficient statistics were kept to minimize memory tasks for counting contingency tables. Komarek and Moore (2000) adapted the static AD-tree to make it computationally tractable for large data sets for machine learning tasks. The dynamic AD-tree developed also had dynamic properties such that it can handle incoming data. HIRED (HIgh-dimensional histogram with dimensionality REDuction) histograms for range querying problems were proposed by Baltrunas et al. (2006), and are constructed by splits along the mid-length of a coordinate where coordinates with higher variation of densities are chosen for splits. This results in a histogram with more bins at regions with higher density variation, thus avoiding unnecessary splits at places with more uniformly distributed data or no data. This reduces the memory requirements needed, especially when compared with regular histograms. Lambert and Pinheiro (2001) proposed a statistical model based on χ^2 tests of independence for the fraud detection problem with a resulting data structure in the form of a set of histograms or "signatures". The method developed is also capable of handling new, incoming data. Gibbons and Matias (1999) studied the performance and accuracy guarantees of "synopsis data structures", a term used to describe data structures that have size significantly smaller than the dataset. Such data structures are small because the data is compressed by summarizing relevant statistics of the data. Most, if not all methods developed have an underlying common principle: developing or exploiting data structures and working with sufficient statistics of the data instead of working directly with the data.

1.2 Statistical Regular Pavings for Massive Data

In this work, we propose a data structure in the form of *statistical regular pavings(SRPs)* to sufficiently summarize massive datasets. An SRP is an extension of a *regular paving (RP)*, which is an ordered binary tree that recursively bisects a box in $\mathbb{R}^d$ along the first widest side. The statistical aspect for an RP is introduced when we allow mutable caches of recursively computable statistics of the data at each node of the RP. We focus on two tasks using this structure: nonparameteric density estimation with histograms, and analyzing spatio-temporal data from real radar data over a busy US airport. In these applications we will emphasise the ability to perform arithmetic efficiently over SRP states to solve various problems.

Density estimation is the task of finding an approximation to an unknown probability density based on observed data. It is fundamental to data analysis and is useful for investigating properties of the given data and for statistical inference. In our study, we focus on nonparametric density estimation, in particular histogram density estimation, and let the data drive the partition of the histogram. Our partitioning scheme is based on the idea of statistically equivalent blocks (SEB) (Gessaman, 1970; Anderson, 1966) where the final partition consists of cells that have at most $\overline{k}_n$ points in them, and is driven by a randomised priority queue. The resulting SRP histogram estimate is asymptotically consistent in the L_1-setting given certain conditions on $\overline{k}_n$.

We are, however, faced with a smoothing problem when trying to choose a suitable $\overline{k}_n$ to produce an optimal histogram from a given collection of n data points. Here, we present a novel method for averaging a sequence of histogram states visited by a Metropolis-Hastings Markov chain whose stationary distribution is the posterior distribution over the space of SRP histograms. The base Markov chain used to propose moves for the Metropolis-Hastings chain is a random walk that data-adaptively prunes and grows the SRP histogram tree. We use a prior distribution based on Catalan numbers and detect convergence heuristically by observing trace plots of suitable summary statistics. SRP states are then sampled from the stationary distribution and we make use of the arithmetic aspect of the SRP structure to obtain an averaged histogram, which is also the posterior mean histogram estimate. Here the averaged histograms are still asymptotically L_1 consistent and are better estimates compared to the states obtained from the randomised priority queue.

The second application of SRPs involves the analysis of high frequency aircraft position information. The main idea is to represent aircraft trajectories by SRP trees such that aggregation/separation can be performed efficiently on the resulting SRP trajectories for various learning tasks. By using aircraft data collected based on weather conditions, we showed that one could easily subtract an SRP trajectory from another SRP trajectory to acquire the difference between the of them. Moreover, through splitting and merging operations, we also showed how the SRP can be transformed into a dynamic structure that allows tracking of flights in the airspace over time and is helpful in memory conservation.

In relation to the massive data problem, we have proposed a data structure that is able to cache sufficient statistics of the data, while preserving the ability to efficiently perform arithmetic operations over the structure, a property that is not readily available in the existing methods discussed above and in the following chapters. Though the space of SRPs is more restrictive compared to other tree-structures that allow for splits at any coordinate and/or at any cut point, byt the restrictive splitting requirement of the SRP allows for a natural recursive property that can be exploited to implement arithmetic operations. Furthermore, by combining different states over the space of the SRPs and whose initial distribution is informed by the randomised priority queue, we are able to visit states that the randomised priority queue might never visit given the observed data.

We will discuss our two major applications in more detail in the following chapters. We first give a formal introduction of RPs and SRPs in Chapter 2 and also explain how recursively computable statistics are cached. Chapter 3 looks at existing nonparametric density estimation methods and then introduces a randomised priority queue based on statistically equivalent blocks to construct SRP histograms that are L_1-consistent. Chapter 4 explains how averaging can be done over SRP histograms and describes a Metropolis-Hastings based algorithm to obtain a posterior mean histogram estimate. Our method is then applied to complicated mixtures of uniform densities over a range of partitions from the space of regular pavings and their L_1 error or integrated absolute error (IAE) for huge sample sizes in large dimensional settings are studied in Chapter 5. We then move on to analyse spatio-temporal datasets in Chapter 6 by using aircraft trajectory data, and will show how SRPs are used to represent trajectories in a manner that allows aggregations to be performed over such trajectories for different learning purposes. Chapter 7 concludes.

Chapter 2

Statistical Regular Paving (SRP)

We describe the regular paving in Section 2.1 and its extension to statistical regular pavings in Section 2.2. The notion of recursively computable statistics is presented in Section 2.2.1.

2.1 Regular Paving (RP)

Let $\boldsymbol{x} := [\underline{x}, \overline{x}]$ be a compact real interval with lower bound $\underline{x}$ and upper bound $\overline{x}$ where $\underline{x} \leq \overline{x}$. Let the space of such intervals be $\mathbb{IR}$. We can then define a box of dimension d as an interval vector

$$\boldsymbol{x} := [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_d, \overline{x}_d] \ .$$

Let $\mathbb{IR}^d$ be the set of all such boxes. Consider a box $\boldsymbol{x} \in \mathbb{IR}^d$. Let the index ι be the first coordinate of maximum width, i.e.

$$\iota = \min \left(\underset{i}{\operatorname{argmax}} (\overline{x}_i - \underline{x}_i) \right) \ .$$

A *bisection* or *split* of $\boldsymbol{x}$ at the mid-point along this first widest component gives us the left and right child boxes of $\boldsymbol{x}$ as follows:

$$\boldsymbol{x}_{\mathsf{L}} := [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_\iota, (\underline{x}_\iota + \overline{x}_\iota)/2) \times [\underline{x}_{\iota+1}, \overline{x}_{\iota+1}] \times \ldots \times [\underline{x}_d, \overline{x}_d] \ ,$$

$$\boldsymbol{x}_{\mathsf{R}} := [\underline{x}_1, \overline{x}_1] \times \ldots \times [(\underline{x}_\iota + \overline{x}_\iota)/2, \overline{x}_\iota] \times [\underline{x}_{\iota+1}, \overline{x}_{\iota+1}] \times \ldots \times [\underline{x}_d, \overline{x}_d] \ .$$

Such a bisection is said to be *regular*. A recursive sequence of selective regular bisections of boxes with possibly open boundaries along the first widest coordinate, starting from the root box $\boldsymbol{x}$ in $\mathbb{IR}^d$ is known as a *regular paving (RP)* (Jaulin et al., 2001) or n-tree (Samet, 1990) of $\boldsymbol{x}$. An RP of $\boldsymbol{x}$ can also be seen as a binary tree formed by recursively bisecting the box $\boldsymbol{x}$ at the root node. In combinatorics this tree is known as the *plane binary tree* (Stanley, 1999, Ex. 6.19(d), p. 220). When the root box $\boldsymbol{x}$ is clear from the context we refer to an RP of $\boldsymbol{x}$ as merely an RP. Each node of an RP is associated with a sub-box of the root box that can be attained by a sequence of selective regular bisections.

Each node in an RP is distinctly labeled by the sequence of child node selections from the root node. We label these nodes and the associated boxes with strings composed of L and R for left and right, respectively. For example, in Figure 2.1, the root node associated with root box $\boldsymbol{x}_\rho$ is labeled ρ. First, we split ρ into two child nodes. These left child and right child nodes are labeled by $\rho\mathsf{L}$ and $\rho\mathsf{R}$, respectively. The left half of $\boldsymbol{x}_\rho$ that is now associated with

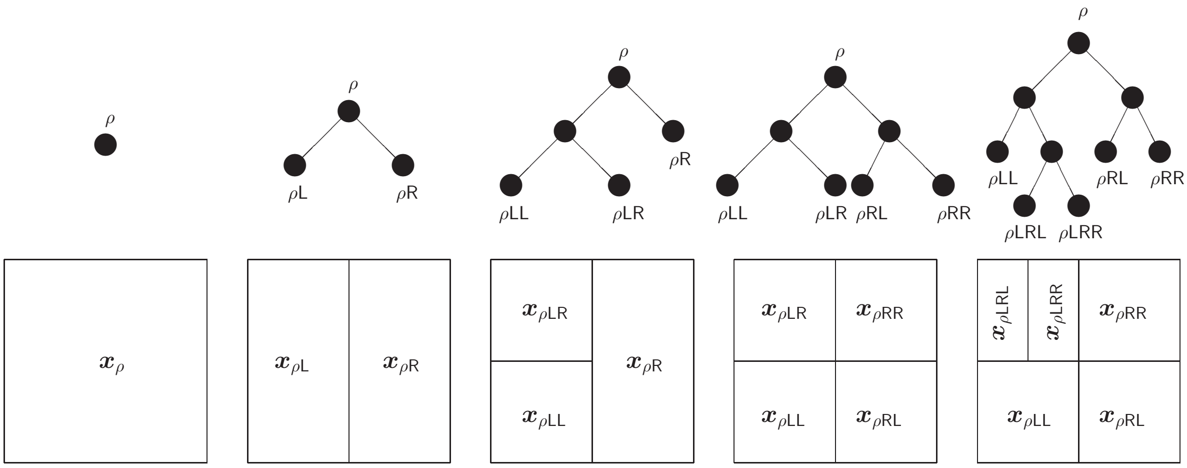


Figure 2.1: A sequence of selective bisections of boxes (nodes) along the first widest coordinate, starting from the root box (root node), produces an RP.

node $\rho\mathsf{L}$ is denoted by $\boldsymbol{x}_{\rho\mathsf{L}}$. Similarly, the right half of $\boldsymbol{x}_\rho$ that is associated with the right child node $\rho\mathsf{R}$ is denoted by $\boldsymbol{x}_{\rho\mathsf{R}}$. We say $\rho\mathsf{L}$ and $\rho\mathsf{R}$ are a pair of *sibling nodes* since they share the same parent node ρ. This pair of sibling nodes can be *reunited* or *merged* to its parent node ρ. A node with no child nodes is called a *leaf node*. A *cherry node* is a sub-terminal node with a pair of leaf nodes that are siblings. These sibling leaf nodes can be reunited or merged back to the cherry node, thereby turning the cherry node into a leaf node in the process. Note that we can only split a leaf node or merge a cherry node. Returning to Figure 2.1, let us further split the left node $\rho\mathsf{L}$ by bisecting the associated box $\boldsymbol{x}_{\rho\mathsf{L}}$ to get its left and right child nodes $\rho\mathsf{LL}$ and $\rho\mathsf{LR}$ with the associated sub-boxes $\boldsymbol{x}_{\rho\mathsf{LL}}$ and $\boldsymbol{x}_{\rho\mathsf{LR}}$, respectively. Next, we split the right child node $\rho\mathsf{R}$ similarly into its child nodes $\rho\mathsf{RL}$ and $\rho\mathsf{RR}$, respectively. Let us select $\rho\mathsf{LR}$ to do a final split and obtain its child nodes $\rho\mathsf{LRL}$ and $\rho\mathsf{LRR}$. We have obtained a binary tree from four splits of the root node. A graphical representation of the obtained RP is shown in Figure 2.1. We denote the label set of all nodes by $\mathbb{V} := \rho \cup \{\rho\{\mathsf{L},\mathsf{R}\}^j : j \in \mathbb{N}\}$.

Let the j-th interval of a box $\boldsymbol{x}_{\rho v}$ be $[\underline{x}_{\rho v,j}, \overline{x}_{\rho v,j}]$. Then the volume of a d-dimensional box $\boldsymbol{x}_{\rho v}$ associated with the node ρv of an RP of $\boldsymbol{x}_\rho$ is the product of the side-lengths of the box, i.e.

$$\mathrm{vol}\,(\boldsymbol{x}_{\rho v}) = \prod_{j=1}^{d} (\overline{x}_{\rho v,j} - \underline{x}_{\rho v,j}) \ .$$

The volume may also be associated with the *depth* of a node. A node has depth k if it can be reached by k splits from the root node. Then, the volume of any d-dimensional box $\boldsymbol{x}_{\rho v}$ associated with node ρv having depth k is $\mathrm{vol}\,(\boldsymbol{x}_{\rho v}) = 2^{-k}\mathrm{vol}\,(\boldsymbol{x}_\rho)$. This is due to the recursive nature of the bisections and the restriction to only bisect perpendicularly at the mid-point along the first widest coordinate. We use the nodes of the RP in Figure 2.1 for illustration purposes. Assume that the root box $\boldsymbol{x}_\rho$ is a unit hypercube. Then the root node ρ has depth 0 and $\mathrm{vol}\,(\boldsymbol{x}_\rho) = 1$, the nodes $\rho\mathsf{L}$ and $\rho\mathsf{R}$ have depth 1 and volume 2^{-1}, the nodes $\rho\mathsf{LL}, \rho\mathsf{LR}, \rho\mathsf{RL}$, $\rho\mathsf{RR}$ have depth 2 and volume 2^{-2}, and finally the nodes $\rho\mathsf{LRL}$, $\rho\mathsf{LRR}$ have depth 3 and volume 2^{-3}.

We can now label each leaf node of a tree by its depth. The leaf nodes of the RP in Figure 2.1, listed in left-right ordering, is $[\rho\mathsf{LL}, \rho\mathsf{LRL}, \rho\mathsf{LRR}, \rho\mathsf{RL}, \rho\mathsf{RR}]$. Then the above RP has 23322 as its *ordered leaf-depth string*. Each RP can be uniquely identified by its ordered leaf-depth string. Thus this RP can be denoted by s_{23322} and the set of leaf boxes associated with its leaf nodes by $\ell(s_{23322}) = \{\boldsymbol{x}_{\rho\mathsf{LL}}, \boldsymbol{x}_{\rho\mathsf{LRL}}, \boldsymbol{x}_{\rho\mathsf{LRR}}, \boldsymbol{x}_{\rho\mathsf{RL}}, \boldsymbol{x}_{\rho\mathsf{RR}}\}$. Among these leaf nodes, we can reunite $\rho\mathsf{LRL}$ and $\rho\mathsf{LRR}$ to get $\rho\mathsf{LR}$, and further reunite $\rho\mathsf{RL}$ and $\rho\mathsf{RR}$ to get $\rho\mathsf{R}$. Note that the nodes $\rho\mathsf{LR}$ and $\rho\mathsf{R}$ of s_{23322} are cherry nodes and the set of boxes associated with its cherry nodes is $c(s_{23322}) = \{\boldsymbol{x}_{\rho\mathsf{LR}}, \boldsymbol{x}_{\rho\mathsf{R}}\}$. Each sequence of splits and merges of an RP with root node ρ returns a partition of its root box $\boldsymbol{x}_\rho$ given by the set of its leaf boxes.

Strictly speaking, the leaf boxes in the RP of Jaulin et al. (2001) do not partition $\boldsymbol{x}$ due to non-empty intersections between the child boxes. Thus the elements of $\ell(s)$ in our RP are not necessarily elements of $\mathbb{IR}^d \cap \boldsymbol{x}$. However, we can always take the interval hull of each leaf box $\boldsymbol{x}_{\rho v}$ in our RP to obtain compact leaf boxes that belong to $\mathbb{IR}^d \cap \boldsymbol{x}$ (Jaulin et al., 2001), if necessary. Note that the interval hull of two intervals $\boldsymbol{a}$ and $\boldsymbol{b}$ is $\boldsymbol{a} \sqcup \boldsymbol{b} = \left[\min\{\underline{a}, \underline{b}\}, \max\{\overline{a}, \overline{b}\}\right]$.

Having seen a particular RP s_{23322} let us study the space of all RPs. Let $\mathbb{S}_k$ be the set of all RPs of $\boldsymbol{x}_\rho$ made of k splits. Note that $|\ell(s)| = k + 1$ if $s \in \mathbb{S}_k$. The number of distinct binary trees with k splits is equal to the Catalan number

$$C_k = \frac{1}{k+1}\binom{2k}{k} = \frac{(2k)!}{(k+1)!(k!)} \quad . \tag{2.1}$$

For $i, j \in \mathbb{Z}_+$, where $\mathbb{Z}_+ := \{0, 1, 2, \ldots\}$ and $i \leq j$, let $\mathbb{S}_{i:j}$ be the set of RPs with k splits where $k \in \{i, i+1, \ldots, j\}$. The space of all RPs is then $\mathbb{S}_{0:\infty} := \lim_{j\to\infty} \mathbb{S}_{0:j}$. Figure 2.2 displays the transition diagram over $\mathbb{S}_{0:3}$ where the gray arrows represent the transition from one RP state to another through a split or reunion. There may be more than one way, i.e. distinct sequence of splits, to reach an RP in $\mathbb{S}_k$ from the root node by applying exactly k splits. Randomised algorithms of interest in the sequel are Markov chains on $\mathbb{S}_{0:\infty}$.

The root node ρ may be thought of as the pointer or address in machine memory space. Then our objective is to enhance the structure of regular pavings via splitting or merging operations in $\mathbb{S}_{0:\infty}$ starting from ρ by means of the data in a possibly adaptive manner for the purpose of nonparametric density estimation in the asymptotic setting (Chapters 3 and 4) or for tracking spatio-temporal data in the most internal-memory efficient manner within $\mathbb{S}_{0:\infty}$ (Chapter 6). The resulting information structure is in the space of statistical regular pavings

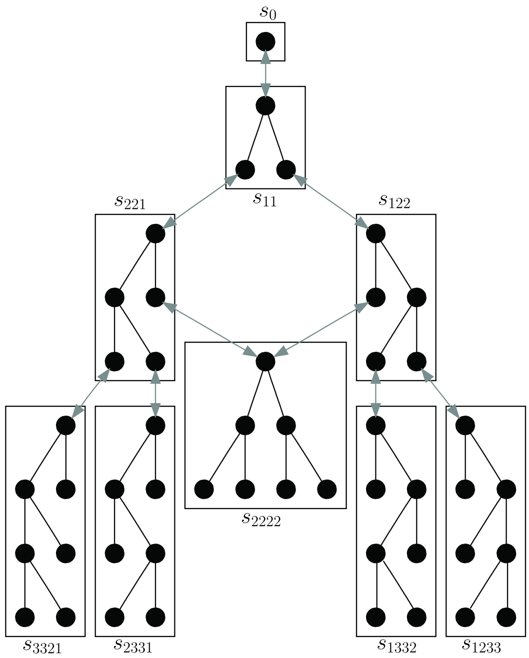


Figure 2.2: Transition diagram over $\mathbb{S}_{0:3}$ with split/reunion transitions from one RP state to another.

or SRPs.

2.2 Statistical Regular Paving (SRP)

Suppose n points $x_1, \ldots, x_n$ where each point x_i is a row vector such that $x_i := (x_{i,1}, \ldots, x_{i,d})$ have fallen into the bounding root box $\boldsymbol{x}_\rho$ of an RP s. We extend the notion of RP to SRP in order to represent a data-driven partition of $\boldsymbol{x}_\rho$. Recall that each node ρv of an RP has a box $\boldsymbol{x}_{\rho v}$ associated with it. We can further associate each node ρv of an RP with *recursively computable statistics* T, such as, (i) $\#\boldsymbol{x}_{\rho v} := \sum_{i=1}^{n} \mathbb{1}_{\boldsymbol{x}_{\rho v}}(x_i)$, the sample count, (ii) the sample mean, (iii) the sample variance-covariance matrix, etc., of the data points that fall into $\boldsymbol{x}_{\rho v}$ for subsequent statistical set processing. Each leaf node has associations via pointers (in our C++ implementation) to the data that lie within its leaf box. When a bisection happens, the data falls into the box associated with either the left or right child node of the bisected node, depending on its location, such that the sample counter in each child node gets recursively updated. Similarly, when two sibling nodes are reunited, the counter of the reunited node remains unchanged as the sum of the counters of the two sibling nodes. Algorithm 1 shows how such statistics T are recursively updated for each data point $x_i \in \mathbb{R}^d$ that has just entered through node ρv. We call this information structure a statistical regular paving (SRP) since it enhances an RP by recording recursively computable statistics of the data for subsequent statistical set processing.

2.2.1 Recursively Computable Statistics

We explain Algorithm 1 in its simplest form as `recalculateStats`$(\rho v, \downarrow^{x_i}, T)$. The simplest recursively computable statistic is the sample count of the data points that have entered through node ρv. Let us denote this by $T^{\boxdot}(\rho v) = \#\boldsymbol{x}_{\rho v} = \sum_{i=1}^{n} \mathbb{1}_{\boldsymbol{x}_{\rho v}}(x_i)$. We can recursively compute $T^{\boxdot}(\rho v)$ at node ρv by substituting $\mathcal{R}_0(x_i) = 0$ and $\mathcal{R}\left(T^{\boxdot}(\rho v), x_i\right) = T^{\boxdot}(\rho v) + 1$ in Algorithm 1 to obtain `recalculateStats`$(\rho v, \downarrow^{x_i}, T^{\boxdot})$. From $T^{\boxdot}(\rho v)$ and $T^{\boxdot}(\rho)$ we can readily compute the empirical measure $\mu_n(\boldsymbol{x}_{\rho v}) := \#\boldsymbol{x}_{\rho v}/n$ over $\boldsymbol{x}_{\rho v}$ from the ratio $T^{\boxdot}(\rho v)/T^{\boxdot}(\rho)$.

As pointed out in Lauritzen (1983), Fisher (1925) observed that the sample mean of the points that enter node ρv, denoted by $\tilde{m}(\rho v) := \#\boldsymbol{x}_{\rho v}{}^{-1} \sum_{i=1}^{n} \mathbb{1}_{\boldsymbol{x}_{\rho v}}(x_i)$, is a recursively computable statistic unlike the sample median. Similarly, the sample variance is also recursively computable (Welford, 1962).

Suppose we are already computing $T^{\boxdot}(\rho v) \in \mathbb{Z}$ and want to further compute the sample mean $\tilde{m}(\rho v) \in \mathbb{R}^{(1\times d)} \cap \boldsymbol{x}_{\rho v}$ of the data points that enter the node ρv. Let $T^{\boxplus}$ be the sample sum vector of all the points that enter node ρv, i.e., $T^{\boxplus}(\rho v) = \sum_{i=1}^{n} x_i, x_i \in \boldsymbol{x}_{\rho v}$. Then we can recursively compute $T(\rho v) = \left(T^{\boxdot}(\rho v), T^{\boxplus}(\rho v)\right)$ by substituting

$$\mathcal{R}_0(x_i) = (0, (0, 0, \ldots, 0)) \in \mathbb{Z} \times \mathbb{R}^{(1\times d)} \quad \text{and}$$
$$\mathcal{R}\left(\left(T^{\boxdot}(\rho v), T^{\boxplus}(\rho v)\right), x_i\right) = \left(T^{\boxdot}(\rho v) + 1, T^{\boxplus} + x_i\right)$$

in Algorithm 1 and obtaining `recalculateStats`$\left(v, \downarrow^{x_i}, \left(T^{\boxdot}, T^{\boxplus}\right)\right)$. Finally, we can compute $\tilde{m}(\rho v)$ from $\left(T^{\boxdot}(\rho v), T^{\boxplus}(\rho v)\right)$ via the ratio $T^{\boxplus}(\rho v)/T^{\boxdot}(\rho v)$.

Now suppose we are already computing $\left(T^{\boxdot}(\rho v), T^{\boxplus}(\rho v)\right)$ and want to further compute $\tilde{\Sigma}(\rho v)$, the sample variance-covariance matrix of the data points that enter node ρv. Let $x_i' \in \mathbb{R}^{(d\times 1)}$ denote the transpose of the data point $x_i \in \mathbb{R}^{1\times d}$ with $x'x \in \mathbb{R}^{d\times d}$ and let $T^{\boxtimes}(\rho v)$ denote the 'sum-product' of the data points in ρv, a precursor statistic for $\tilde{\Sigma}(\rho v)$. Then we can recursively compute $T(\rho v) = \left(T^{\boxdot}(\rho v), T^{\boxplus}(\rho v), T^{\boxtimes}(\rho v)\right)$ by substituting:

$$\mathcal{R}_0(x_i) = \left(0, (0, 0, \ldots, 0), \begin{pmatrix} 0 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & 0 \end{pmatrix}\right) \in \mathbb{Z} \times \mathbb{R}^{(1\times d)} \times \mathbb{R}^{(d\times d)}, \quad \text{and}$$
$$\mathcal{R}\left(\left(T^{\boxdot}(\rho v), T^{\boxplus}(\rho v), T^{\boxtimes}(\rho v)\right), x_i\right) = \left(T^{\boxdot}(\rho v) + 1, T^{\boxplus} + x_i, T^{\boxtimes} + x_i' x_i\right)$$

in Algorithm 1 and obtaining `recalculateStats`$\left(\rho v, \downarrow^{x_i}, \left(T^{\boxdot}, T^{\boxplus}, T^{\boxtimes}\right)\right)$. Finally, we can compute a non-zero $\tilde{\Sigma}(\rho v)$ from $\left(T^{\square}(\rho v), T^{\boxplus}(\rho v), T^{\boxtimes}(\rho v)\right)$, provided $T^{\square}(\rho v) > 1$, as follows:

$$\tilde{\Sigma}(\rho v) = \left(T^{\square}(\rho v)\left(T^{\square}(\rho v) - 1\right)\right)^{-1}\left(T^{\square}(\rho v)T^{\boxtimes}(\rho v) - \left(T^{\boxplus}(\rho v)\right)' T^{\boxplus}(\rho v)\right) .$$

These algorithms are implemented using `dotprecision` data type from the C-XSC library (Wedner and Hofschuster, 2001) since values and variables of these data types are exactly representable in this format, i.e. without rounding error, independent of the size of the vectors or matrices contained in the scalar product expressions, and thereby permit the exact summation of an arbitrary number of such products in a dotprecision accumulator with maximum precision.

We finally introduce one of the most useful box-valued recursively computable statistics,

$$T^{\square}(\rho v) = \left(T_1^{\square}(\rho v), \ldots, T_d^{\square}(\rho v)\right) = \left(\left[\underline{T}_1^{\square}(\rho v), \overline{T}_1^{\square}(\rho v)\right], \ldots, \left[\underline{T}_d^{\square}(\rho v), \overline{T}_d^{\square}(\rho v)\right]\right) ,$$

that takes values in $\mathbb{IR}^d$ and updates the convex hull of the data points that have fallen through node ρv. We can recursively compute $T^{\square}(\rho v)$ by substituting

$$\begin{aligned}
&\mathcal{R}_0(x_i) = ([+\infty, -\infty], \ldots, [+\infty, -\infty]) \notin \mathbb{IR}^d, \text{ and} \\
&\qquad \mathcal{R}\left(T^{\square}(\rho v), x_i\right) = \left[\mathcal{R}\left(T_1^{\square}(\rho v), x_{i,1}\right), \ldots, \mathcal{R}\left(T_d^{\square}(\rho v), x_{i,d}\right)\right], \text{ where,} \\
&\qquad \mathcal{R}\left(T_j^{\square}(\rho v), x_{i,j}\right) = \left[\inf\left(\underline{T}_j^{\square}(\rho v), x_{i,j}\right), \sup\left(\overline{T}_j^{\square}(\rho v), x_{i,j}\right)\right], \quad \text{for } j = 1, \ldots, d
\end{aligned}$$

in Algorithm 1 and obtaining `recalculateStats`$(\rho v, \downarrow^{x_i}, T^{\square})$.

Now, when a node bisection happens, `nodeExpand` of Algorithm 3 is called to insert the data into the child nodes by first calling `insertOneFind` of Algorithm 2, which in turns recursively update the statistics in each child node by calling `recalculateStats`.

Figure 2.3(a) depicts the details of an SRP tree with its nodes and associated leaf boxes forming the partition of the root box along with their associations with the data (gray arrows). Once the SRP has been constructed, the associations to data from the leaf nodes are removed and only the height statistics at all nodes remain, resulting in an *SRP histogram* as depicted in Figure 2.3(b). This is a significant memory-saving intermediary step when comparing batches of massive simulated data during simulation-intensive inference problems or when producing a sequence of density estimates from pulses of massive real-world data (Chapter 6). In this setting, the root box remains the same, assuming that we already have prior information regarding the support set of the data. The partition of the root box is adapted to the bursts of data that enter into the root box and the recursively computable statistics at each node are updated accordingly. By an abuse of notation, we denote an RP as well as an SRP by s and the space of all RPs as well as SRPs by $\mathbb{S}_{0:\infty}$.

Algorithm 1: `recalculateStats` $(\rho v, \downarrow^{x_i}, T)$

input :

- node: ρv
- pointer: $\downarrow^{x_i}$ to new data point $x_i \in \mathbb{R}^d$ through node ρv
- recursively computable statistics T to be updated at node ρv

action : update $T(\rho v)$, the recursively computable statistics T at node ρv

```
if T(ρv) = ∅ then
    T(ρv) ← R_0(x_i);            // initialize T from first data point through ρv
end
T(ρv) ← R(T(ρv), x_i);          // recursively update T at node ρv
```

Algorithm 2: `insertOneFind` $(\rho v, \downarrow^{x_i}, T)$

input :

1. node: ρv
2. pointer: $\downarrow^{x_i}$ to data point $x_i \in \mathbb{R}^d$
3. recursively computable statistics: T

output: Boolean: $\mathfrak{b}$

```
b = false

if (x_i ∈ x_ρv) then
    recalculateStats(ρv, ↓^{x_i}, T)
    if IsLeaf(ρv) then
        ρv_[↓].append(↓^{x_i}) ;   // append ↓^{x_i} to ρv's data pointer list ρv_[↓]
        b = true
    end
    else
        b = insertOneFind(ρvL, ↓^{x_i}, T)
        if (b = false) then
            b = insertOneFind(ρvR, ↓^{x_i}, T)
        end
    end
end
return b
```

Algorithm 3: `nodeExpand` $\left(\rho v, \rho v_{[\downarrow]}, T, \right)$

input :

1. node: ρv for bisection
2. data pointer for node ρv: $\rho v_{[\downarrow]}$
3. recursively computable statistics: T

output: Boolean: $\mathfrak{b}$

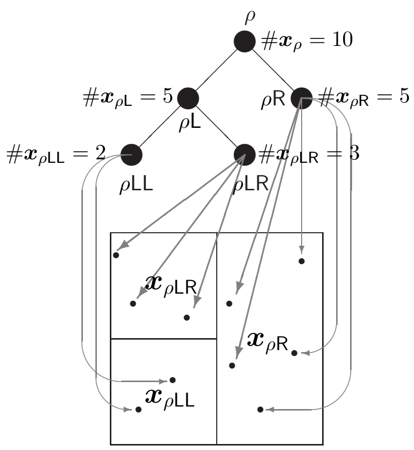
$\mathfrak{b} = \texttt{false}$

Make left node $\rho v\mathsf{L}$ with box $\boldsymbol{x}_{\rho v\mathsf{L}}$ and graft onto ρv as left child the node $\rho v\mathsf{L}$
Make right node $\rho v\mathsf{R}$ with box $\boldsymbol{x}_{\rho v\mathsf{R}}$ and graft onto ρv as right child the node $\rho v\mathsf{R}$
$\rho v\mathsf{L}_{[\downarrow]} \leftarrow [], \rho v\mathsf{R}_{[\downarrow]} \leftarrow []$; `// Initalise the list of data pointers for nodes` $\rho v\mathsf{L}$ `and` $\rho v\mathsf{R}$`.`

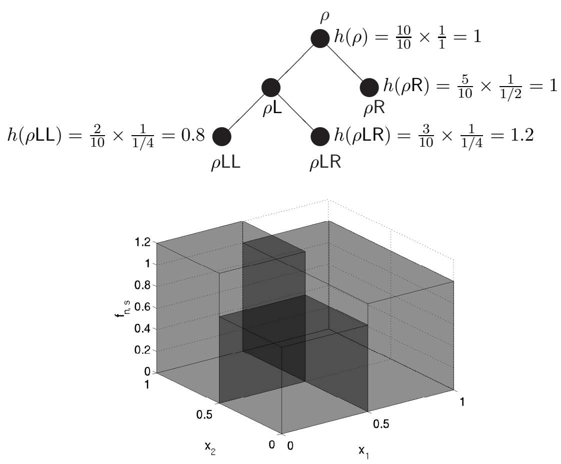
foreach $(\downarrow^{x_i} \in \rho v_{[\downarrow]})$ **do**
  $\mathfrak{b} = \texttt{insertOneFind}(\rho v\mathsf{L}, \downarrow^{x_i}, T)$
  if $(\mathfrak{b} = \texttt{false})$ **then**
    $\mathfrak{b} = \texttt{insertOneFind}(\rho v\mathsf{R}, \downarrow^{x_i}, T)$
  end
end

$\rho v_{[\downarrow]}$`.clear`; `// Clear the data pointers of this node.`

return $\mathfrak{b}$



(a) An SRP tree and its constituents.



(b) An SRP histogram and its tree.

Figure 2.3: An SRP and its corresponding histogram.

Chapter 3

Adaptive Histograms from Randomized Priority Queues

3.1 Histogram Density Estimation

Suppose $X_1, \ldots, X_n$ are independent and identical random vectors in $\mathbb{R}^d$, each distributed according to μ and having a non-atomic density $f \in L_1(\lambda)$, i.e. $P(X_1 \in A) = \mu(A) = \int_A f d\lambda$, where μ is absolutely continuous with respect to d-dimensional Lebesgue measure λ or $\mu \ll \lambda$. We are interested in the task of estimating f by histogram density estimation. Histogram estimators are nonparametric and are able to adapt to the data without any underlying assumptions.

Wand (1997) and Birgé and Rozenholc (2006) provided major results in getting the optimal partition in the L_2-setting for the classical regular histogram. However, regular histograms can be problematic. Rissanen et al. (1992) argued that in low density regions, it was unnecessary to have large amount of bins and resources should instead be concentrated on places of high densities, especially if the underlying distribution is non-uniform. Klemelä (2009) showed concern regarding dimensionality issues especially when performing density estimation using regular histograms as such histograms "are not able to adapt to spatially varying smoothness". He proposed that the partitions should be allowed to be chosen in a flexible way. For example, have thinner bins at varying density regions, especially in high-dimensional cases to get around the curse of dimensionality. Scott (1985) introduced averaged shifted histograms and the frequency polygon for averaged shifted histograms for analysis up to 4 dimensions in an attempt to overcome the problem of fixed-width histograms. The main idea in constructing an averaged shifted histogram was to take the average over m histograms with equal bin widths h but different bin origins; whilst the frequency polygon for the averaged shifted histogram was constructed by connecting the mid-points of the bins using piecewise linear functions. Hearne and Wegman (1994), however, suggested that "the ASH algorithm is computationally intensive" and also proposed a density estimator based on random-width bins by "maintaining a minimum number of samples in each bin" (Scott and Sain, 2005) using a random sub-sampling scheme and optimizing the likelihood function. However this method was also computationally intensive (Chen and Kelton, 2006).

The problem of constructing irregular histograms is more complicated than finding an optimal bin width and starting point for a regular histogram as it involves finding an optimal set of cut points in addition to the number of bins. Furthermore, rather than using a fixed sequence of partitions, Stone (1982), as cited in Lugosi and Nobel (1996), promoted the use of data-dependent partitions and supplied theoretical evidence that data-driven histogram estimators perform better.

There have been various approaches developed for irregular histogram estimation. In the L_2-setting, Rudemo (1982) used cross-validation for histograms and Celisse and Robin (2008) proposed explicit formulas for leave-p-out cross validation for regular and irregular histograms. However, Devroye and Lugosi (2004) argued that cross-validation techniques will not work for densities with especially large peaks, and could lead to non-consistency. Bin choices based on asymptotic results were studied by Kogure (1987) and Kanazawa (1992). The method proposed by Kogure (1987) was dependent on tuning parameters and was not an automated procedure. Kanazawa (1992) used the Hellinger distance as the optimality criterion but his approach required the first and second derivatives of the unknown density, which is impractical to apply.

Penalized likelihood methods are also common and often based on the Akaike information criterion (AIC) (Akaike, 1974). Taylor (1987) minimized AIC to derive the asymptotically optimal bin width for 1-dimensional data. Birgé and Rozenholc (2006) used a non-asymptotic evaluation of the performances of penalized maximum likelihood estimator in some exponential families due to Castellan (1999) and intensive simulations to optimize the form of the penalty function. These methods were data-based partitioning schemes for regular histograms. Hartigan (1996) then compared equal-bin-width “Akaike-histograms” to Bayesian histograms constructed with a subjective smoothing parameter to control the number of elements in his partitions. Castellan (1999) extended this to multivariate data and irregular histograms. Rozenholc et al. (2009, 2010) suggested that “irregular partitions can reduce bias” but getting an optimal partition for irregular histograms may be difficult and could “lead to an increase in risk for more well-behaved densities”. Their proposed method was data-driven and automated, and selected either a regular or irregular histogram constructed based on the maximum penalized log-likelihood. They considered risk functions in the form of the squared Hellinger distance, and the L_1- and L_2-norms. The Hellinger criterion was used to choose a suitable penalty function motivated from work by Castellan (1999), Barron et al. (1999), and Massart (2007). Rozenholc et al. (2009, 2010) also compared their method with the taut string procedure of Davies and Kovac (2004), which focused on finding a density estimate with minimum modes and is capable of locating density peaks, to generate irregular histograms. A Kuiper metric that considers the differences in probability over disjoint intervals was used to better detect modality. Some of the problems with the complexity penalized approach were discussed in Birgé and Rozenholc (2006). This methodology was based on the asymptotically optimal performance of penalized maximum likelihood estimators but was constrained by the best form of the penalty function itself being dependent on the unknown underlying density.

Rissanen et al. (1992) and Kontkanen and Myllymki (2007) introduced an information theoretic framework for histogram density estimation based on a minimum description length principle. The minimum description principle exploits the regularity in data as the presence of such regularity allows for data compression. The main idea in both Rissanen et al. (1992) and Kontkanen and Myllymki (2007) was to find a description that will describe the data in an optimum way. Rozenholc et al. (2010) argued that Rissanen et al. (1992)'s method was expensive as it required an exhaustive search over all possible combinations of parameter values. Kontkanen and Myllymki (2007) considered histograms of unequal bin width and treated histogram density estimation as a model selection problem where cut point sets were considered. The optimal set of cut points and globally optimum number of bins were found via a dynamic algorithm that minimizes a stochastic complexity criterion based on the normalized maximum likelihood distribution.

All methods to find the optimal histogram for a given sample size n described above were for the univariate setting. Tree-based approaches were developed for density estimation and can be used in the multivariate setting. Density estimation using dyadic partitions that allow splits at the midpoint at any coordinate of any node were studied by Engel (1997), Blanchard et al. (2007) and Klemelä (2007, 2009), with optimization criterion based on the L_2-error. Dyadic trees have nodes that are split at its midpoint without any restrictions to the splitting direction so that the choice of partitions is flexible. Engel (1997) introduced multiresolution histograms which involved parameter tuning, but no universal recommendations were given. Blanchard et al. (2007)'s method required exhaustive searches over a restricted set of partitions. Klemelä (2007) discussed a CART-like (Breiman et al., 1984) methodology for density estimation, which involved partitioning using a greedy algorithm to minimize an empirically-approximated L_2-based error followed by pruning to minimize the complexity-penalized error. Klemelä (2009) described adaptive density estimation with best basis selection for multi-dimensional data by determining the dimension on which to subdivide some element of the partition. This algorithm grows the tree by bisecting each element successively on each possible dimension until a specified maximum number of bisections in each dimension has been reached, followed by pruning to minimize a complexity penalized L_2 error. However, all these methods required exhaustive searches over a potentially large space of partitions, and are unable to computationally cope with massive data in high dimensions.

3.1.1 SRP Histograms

Our goal is to construct multivariate SRP histogram estimates $f_{n,s}$ (e.g. Figure 2.3(b) of Chapter 2) that are data-driven and able to cope with massive data. We also require the histogram estimates to be L_1-consistent asymptotically, i.e.

$$\int |f(x) - f_{n,s}(x)|\, dx \to 0$$

with probability 1 as $n \to \infty$. One such consistent method is the $k-$spacing density estimate for univariate densities or estimators based on SEB. Here the real line is partitioned into intervals such that each interval, with the possible exception of the rightmost, contains k_n points. The k-spacing partitioning scheme can be generalized in several ways for multivariate data as seen in the work of Gessaman (1970) and Anderson (1966). These partitioning schemes require each cell to have exactly k_n points (possibly excluding the rightmost set). Gessaman (1970) obtained m^d rectangles by projecting, in order, from the first coordinate axis to the last, and partitioning the data into m sets using hyperplanes perpendicular to the projected axis, where $m = \lceil (n/k_n)^{1/d} \rceil$. A more flexible version of the $k-$spacing rule for classification was presented in Devroye et al. (1996, Theorem 21.3, p. 373) where block equivalence is allowed up to an interval $\{\mathbf{k}_n = [\underline{k}_n, \overline{k}_n]\}$.

In the subsequent sections we will describe the construction of a SRP histogram using a partitioning scheme driven by a randomised priority queue based on the generalized SEB, i.e. we allow each leaf node to have at most $\overline{k}_n$ number of points. Section 3.2 will introduce the priority queue that determines which node should be selected for splitting via an appropriate comparison function. No further splits are allowed when all the leaf nodes in a current state has at most $\overline{k}_n$ points and this state will be our SRP histogram constructed using the SEB criterion. However, without some constraints on the relative growth rate of $\overline{k}_n$ we cannot hope to get a strongly L_1 consistent partitioning scheme. Hence we will look at sufficient conditions that are required for L_1-consistency in Section 3.3. Once a histogram estimate is constructed, we would like to ensure that it is computationally efficient to further process information for subsequent inference tasks. Memory and time complexities are discussed in Section 3.4 for the post-constructive setting and we will show that our histogram estimators need no more than $O(nd)$ space and time requirements.

We now have an L_1 consistent partitioning scheme but we are faced with the "smoothing problem". Figure 3.1 shows two different SRP histograms constructed using two different values of $\overline{k}_n$ for the same dataset of 10^5 points simulated under the standard bivariate Gaussian density. A small $\overline{k}_n$ produces histogram that is under-smoothed with unnecessary spikes (left) while the other histogram with a larger $\overline{k}_n$ used as the SEB criterion is over-smoothed (right).

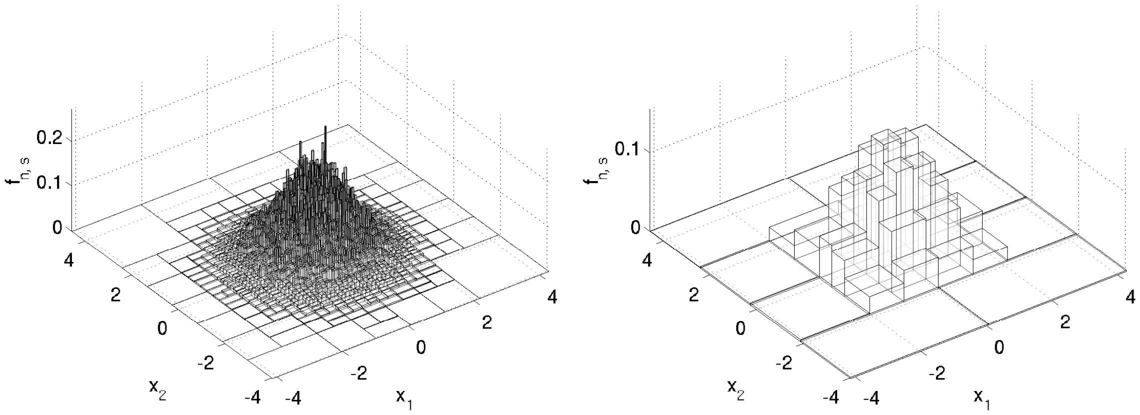


Figure 3.1: Two histogram density estimates for the standard bivariate gaussian density. The left figure shows a histogram with 1485 leaf nodes where $\overline{k}_n = 50$ and the histogram on the right has $\overline{k}_n = 1500$ resulting in 104 leaf nodes.

Our approach to solve the smoothing problem is to estimate the expectation of the posterior probability distribution over a set of partitions from the average of a number of independent random samples of partition states from the distribution. Though our SRP tree structure is similar to both the dyadic trees in the sense that the nodes of our SRP tree retain statistics of the data points contained within, our tree structures have a more restricted state space. The dyadic partitioning scheme allows for splitting in any direction whilst the SRP space is restricted to partitions with splits at the midpoint of the first coordinate with the largest width or volume, but is closed under addition and scalar multiplication. The restricted but algebraic structure of the SRP space allows us to propose an efficient algorithm for averaging multi-dimensional histograms with regularly paved partitions and thereby obtain the sample posterior mean of such multivariate histograms. We construct a Metropolis-Hastings Markov chain under a Catalan prior distribution to produce the sample mean estimate of the Bayesian posterior expectation over this space of histograms (see Chapter 4). We will show using simulations in Chapter 5 that the posterior mean histogram has lower L_1 errors than the optimal SRP histogram one can obtain from the set of all histogram states that can be visited by our asymptotically L_1 randomised priority queue.

3.2 Randomised Priority Queues for Recursively Computable Statistics

We first need to construct an appropriate root box $\boldsymbol{x}_\rho$ of our initial SRP $s = s_0$ with the initial leaf and root node ρ before describing how the histogram is constructed adaptively. Let $\mathbf{x} := \{x_1, x_2, \ldots, x_n\}$ be the support set of n data points $x_1, x_2, \ldots, x_n$. An appropriate root box $\boldsymbol{x}_\rho$ is constructed as each of the data points sequentially enter ρ in one burst. The root box $\boldsymbol{x}_\rho$ can be determined with or without knowledge about the data. If useful prior information about $\mathbf{x}$ is known and $\mathbf{x} \in \mathbb{IR}^d$, the root box $\boldsymbol{x}_\rho$ is just the support set $\mathbf{x}$. Otherwise, we obtain a data-dependent root box from the convex hull $T^{\square}$ of the data points that have fallen through ρ, with an appropriate padding $\xi > 0$ for the box. By the end of the burst, our root box construction procedure should yield a satisfactory $\boldsymbol{x}_\rho$, with the desired statistics including $\boldsymbol{T}^{\boxdot}$ recursively updated for each data point that entered into the root box. We describe this in `makeBox` of Algorithm 4.

We can now proceed to construct an SRP histogram using a randomised algorithm that chooses a node for splitting according to some condition.

Let $\psi \circ (\mathcal{R}_i(T(\rho v), x_i)) := \psi \circ \mathcal{R}(\rho v)$ be a priority function for a node ρv. As data arrive, the leaf boxes are ordered by their priority functions such that the leaf box with the largest $\psi \circ \mathcal{R}$ value is chosen for the next bisection. Thus, the current set of leaf nodes and thereby $\ell(s)$, the associated leaf boxes, of our current SRP s is maintained as a queue that is prioritized by the function $\psi \circ \mathcal{R}$.

Now let $\hat{\ell}(s)$ denote the subset of all leaf boxes in the SRP s which have equally the largest $\psi \circ \mathcal{R}$ value. If there are two or more nodes in $\hat{\ell}(s)$ we break ties by picking these boxes at random for the next bisection. This results in $\texttt{RPQ}(\psi \circ \mathcal{R})$, a *randomised priority queue* for $\psi \circ \mathcal{R}$ that maintains the leaf nodes of the current SRP for the next bisection. Once a leaf node is removed and bisected its two child nodes are inserted as new leaf nodes into the $\texttt{RPQ}(\psi \circ \mathcal{R})$. In general, we remove boxes that are devoid of data points from the queue to avoid unnecessary splits and save memory, especially in high dimensions.

This randomised priority queue for ψ or `RPQ` is listed in Algorithm 5. The path of `RPQ` yields a discrete-time Markov chain $\{S(i)\}_{i=0}^{\dot{I}}$ on $\mathbb{S}_{0:\infty}$. The Markov chain $\{S(i)\}_{i=0}^{\dot{I}}$ stops at a terminal SRP $\dot{S} := S(\dot{I})$ to produce a random partition $\ell(\dot{S}) \cup \boldsymbol{x}_\rho^c$ when each leaf box $\boldsymbol{x}_{\rho v} \in \ell(\dot{S})$ contains at most $\overline{k}_n$ points or $|\ell(\dot{S})| = \overline{m}_n$. For a particular data sample $x_1, \ldots, x_n$ where each point x_i is a row vector such that $x_i = (x_{i,1}, \ldots, x_{i,d})$, we get a realization of $\dot{S} = \dot{s} \in \mathbb{S}_{0:\infty}$ with partition $\ell(\dot{s}) \cup \boldsymbol{x}_\rho^c$. For each leaf box $\boldsymbol{x}_{\rho v} \in \ell(\dot{s})$, let $\mu_n(\boldsymbol{x}_{\rho v}) := \#\boldsymbol{x}_{\rho v}/n$ be its empirical measure based on the proportion of sample points $x_1, \ldots, x_n$ that fell into it and let $\lambda(\boldsymbol{x}_{\rho v}) := \mathrm{vol}\,(\boldsymbol{x}_{\rho v})$ be its Lebesgue measure in $\mathbb{R}^d$. Let $\boldsymbol{x}(x)$ be the leaf box containing

Algorithm 4: `makeBox`$(\rho, [\downharpoonleft^{x_1}, \ldots, \downharpoonleft^{x_n}], T, \mathbf{x})$

input :

1. root node: ρ;
2. pointers: $[\downharpoonleft^{x_1}, \ldots, \downharpoonleft^{x_n}]$ to a sequential stream of data points: $[x_1, \ldots, x_n]$;
3. recursively computable statistics: T, with $T^{\boxdot} \in T$;
4. support set of data $\{x_1, \ldots, x_n\}$: $\mathbf{x}$ with the following cases:-
   (a) $\mathbf{x} \in \mathbb{IR}^d$, if useful prior information is available and $\mathbf{x}$ is a known box in $\mathbb{IR}^d$;
   (b) $\mathbf{x} = ([\emptyset], \xi)$, if no useful prior information is available and so we get a data-dependent root box from $T^{\square} \in T$ with padding parameter $\xi > 0$.

action : construct a root box $\boldsymbol{x}_\rho \in \mathbb{IR}^d \cup \{[\emptyset]\}$

$\rho_{[\downharpoonleft]} \leftarrow [\,]$; `//` $\rho_{[\downharpoonleft]}$`, the list of data pointers from` ρ `is initially empty`

for $i = 1 : n$ **do**

  `recalculateStats`$(\rho, \downharpoonleft^{x_i}, T)$; `// recursively update the statistics in` ρ

  $\rho_{[\downharpoonleft]}$`.append`$(\downharpoonleft^{x_i})$; `// append` $\downharpoonleft^{x_i}$`, the pointer to` x_i`, to` $\rho_{[\downharpoonleft]}$

end

if $\mathbf{x} = ([\emptyset], \xi)$ **then**

  $\boldsymbol{x}_\rho \leftarrow T^{\square}(\rho) \pm \xi/n$; `// the data-dependent root box with padding`

end

else if $\mathbf{x} \in \mathbb{IR}^d$ **then**

  $\boldsymbol{x}_\rho \leftarrow \mathbf{x}$; `// the root box is the given support`

end

return $\boldsymbol{x}_\rho$

x. Then the histogram density estimate based on the terminal output SRP $\dot{s}$ of the Markov chain $\{S(i)\}_{i=0}^{\dot{I}}$ associated with `RPQ` of Algorithm 5 is defined by

$$f_{n,\dot{s}}(x) = \begin{cases} \frac{\mu_n(\boldsymbol{x}(x))}{\lambda(\boldsymbol{x}(x))}, & \text{if } 0 < \lambda(\boldsymbol{x}(x)) < \infty, \boldsymbol{x}(x) \in \ell(\dot{s}), \\ 0, & \text{otherwise.} \end{cases}$$

Note that this histogram estimate is the maximum likelihood estimate over all simple functions that integrate to 1 over the partition given by $\ell(\dot{s})$.

3.2.1 Statistics for Node Comparisons

In our `RPQ` the priority function $\psi \circ \mathcal{R}(\rho v)$ allows us to compare any two leaf nodes to determine their priority in the randomised queue of all current leaf nodes for the next bisection. As $S(i+1)$ is obtained from $S(i)$ in `RPQ` by bisecting a leaf box $\boldsymbol{x}_{\rho v}$ in $\hat{\ell}(S(i))$ at random, we can recursively update the statistics needed to compute $\psi \circ \mathcal{R}$ at the new leaf nodes of $S(i+1)$ with boxes $\boldsymbol{x}_{\rho v \mathsf{L}}$ and $\boldsymbol{x}_{\rho v \mathsf{R}}$. Suppose ρu and ρv are two leaf nodes whose priorities are given by comparing $\psi \circ \mathcal{R}(\rho u)$ and $\psi \circ \mathcal{R}(\rho v)$. Their priorities are determined in terms of the magnitude of the priority function $\psi \circ \mathcal{R}$. Here the leaf node comparison criterion `CompCount` by simply comparing counts with $\psi \circ \mathcal{R}(\rho v) = \#\boldsymbol{x}_{\rho v}$ is used. This criterion focuses on areas with higher sample counts, and thereby produces histograms with more information at places with higher densities. Regions with lower densities are less visited, avoiding unnecessary splits at places with less points. The histogram produced will thus have varying bin widths. This is particularly useful in higher-dimensional settings as priority is given to boxes where the underlying density is concentrated at in terms of resources available, while sparser regions which may not have any useful information will be least visited by the queue. This is unlike the regular histogram for which bin widths of equal size are required and hence the splits are spreadly uniformly over the support, even at sparse regions. This is a waste of resources in particularly memory space, and especially if we only want to focus on the more informative regions.

3.3 Asymptotic Consistency

We show that our adaptive histogram based on `RPQ` is consistent by proving the three conditions in Theorem 1 of Lugosi and Nobel (1996), i.e.

(a) the number of leaf boxes grows sub-linearly;

(b) the partition grows sub-exponentially in terms of a combinatorial complexity measure;

(c) and the volume of the leaf boxes in the partition are shrinking,

Algorithm 5: `RPQ` $\left(\boldsymbol{x}_\rho, \rho_{[\downarrow]}, \psi \circ \mathcal{R}, \overline{k}_n, \overline{m}_n\right)$

input :

1. root box obtained from `makeBox`: $\boldsymbol{x}_\rho$;
2. data pointer to node ρ: $\rho_{[\downarrow]}$;
3. priority function: $\psi \circ \mathcal{R}$;
4. SEB max: $\overline{k}_n = n^\alpha$;
5. maximum partition size: $\overline{m}_n$.

output : histogram estimate $f_{n,s}$.

initialize:

`nodeExpand` $\left(\rho, \rho_{[\downarrow]}, T\right)$

$q \leftarrow \emptyset$; `// initialise the queue sorted by` $\psi \circ \mathcal{R}$ `where the largest value is in front`

if $(\psi \circ \mathcal{R}(\rho\mathsf{L}) = \psi \circ \mathcal{R}(\rho\mathsf{R})\ \ \&\ \ \#\boldsymbol{x}_{\rho\mathsf{L}} \neq 0\ \ \&\ \ \#\boldsymbol{x}_{\rho\mathsf{R}} \neq 0)$ **then**

  insert $\rho\mathsf{L}$ and $\rho\mathsf{R}$ into the set $\hat{\ell}(s)$

  q.append($\hat{\ell}(s)$) ; `// insert into queue`

end

else if $(\psi \circ \mathcal{R}(\rho\mathsf{L}) > \psi \circ \mathcal{R}(\rho\mathsf{R})\ \ \&\ \ \#\boldsymbol{x}_{\rho\mathsf{L}} \neq 0)$ **then**

  insert $\rho\mathsf{L}$ into the set $\hat{\ell}(s)$

  q.append($\hat{\ell}(s)$) ; `// insert into queue`

  q.append($\rho\mathsf{R}$) if $\#\boldsymbol{x}_{\rho\mathsf{R}} \neq 0$; `// next-in-line`

end

else

  $(\psi \circ \mathcal{R}(\rho\mathsf{R}) > \psi \circ \mathcal{L}(\rho\mathsf{R})\ \ \&\ \ \#\boldsymbol{x}_{\rho\mathsf{R}} \neq 0)$

end

insert $\rho\mathsf{R}$ into the set $\hat{\ell}(s)$ q.append($\hat{\ell}(s)$) ; `// insert into queue`

q.append($\rho\mathsf{L}$) if $\#\boldsymbol{x}_{\rho\mathsf{L}} \neq 0$; `// next-in-line`

repeat

  $\boldsymbol{x}_{\rho v^*} \leftarrow$ Uniform($\hat{\ell}(s)$) ; `// sample uniformly at random from` $\hat{\ell}(s)$

  q.pop($\boldsymbol{x}_{\rho v^*}$) ; `// remove this node from the queue`

  `nodeExpand` $\left(\boldsymbol{x}_{\rho v^*}, \rho v^*_{[\downarrow]}, T,\right)$

  `// insert child nodes into the appropriate position of the queue`

  q.append($\rho v^*\mathsf{L}$) if $\#\boldsymbol{x}_{\rho v^* \mathsf{L}} \neq 0$

  q.append($\rho v^*\mathsf{R}$) if $\#\boldsymbol{x}_{\rho v^* \mathsf{R}} \neq 0$

  update $\hat{\ell}(s)$

until $\#\boldsymbol{x}_{\rho v} \leq \overline{k}_n$ *for each* $\boldsymbol{x}_{\rho v} \in \ell(s)$ *and* $|\ell(s)| \leq \overline{m}_n$;

return $f_{n,s}$

as the number of sample points increases linearly. Before proceeding to Proposition 1, we need some definitions. Let $\{S_n(i)\}_{i=0}^{\dot{I}}$ on $\mathbb{S}_{0:\infty}$ be the Markov chain formed using `RPQ` of Algorithm 5. The Markov chain terminates at some state $\dot{s}$ with partition $\ell(\dot{s})$. Associated with the Markov chain is a fixed collection of partitions

$$\mathcal{L}_n := \left\{ \ell(\dot{s}) : \dot{s} \in \mathbb{S}_{0:\infty}, Pr\{S(\dot{I}) = \dot{s}\} > 0 \right\}$$

and the size of the largest partition $\ell(\dot{s})$ in $\mathcal{L}_n$ is given by

$$m(\mathcal{L}_n) := \sup_{\ell(\dot{s}) \in \mathcal{L}_n} |\ell(\dot{s})| \leq \overline{m}_n$$

such that $\mathcal{L}_n \subseteq \{\ell(s) : s \in \mathbb{S}_{0:\overline{m}_n - 1}\}$.

Given n fixed points $\{X_1, \ldots, X_n\} \in \left(\mathbb{R}^d\right)^n$. Let $\Pi\left(\mathcal{L}_n, \{X_1, \ldots, X_n\}\right)$ be the number of distinct partitions of the finite set $\{X_1, \ldots, X_n\}$ that are induced by partitions $\ell(\dot{s}) \in \mathcal{L}_n$:

$$\Pi(\mathcal{L}_n, \{X_1, \ldots, X_n\}) := |\{\{\boldsymbol{x}_{\rho v} \cap \{X_1, \ldots, X_n\} : \boldsymbol{x}_{\rho v} \in \ell(\dot{s})\} \ : \ell(\dot{s}) \in \mathcal{L}_n\}| \ .$$

For any fixed set of n points, the growth function of $\mathcal{L}_n$ is then

$$\Pi^*(\mathcal{L}_n, \{X_1, \ldots, X_n\}) = \max_{\{X_1, \ldots, X_n\} \in \left(\mathbb{R}^d\right)^n} \Pi(\mathcal{L}_n, \{X_1, \ldots, X_n\}) \ .$$

Let $A \subseteq \mathbb{R}^d$. Then the diameter of A is the maximum Euclidean distance between any two points of A, i.e. $\mathrm{diam}(A) := \sup_{x,y \in A} \sqrt{\sum_{i=1}^d (x_i - y_i)^2}$. Thus, for a box $\boldsymbol{x} = [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_d, \overline{x}_d]$, $\mathrm{diam}(\boldsymbol{x}) = \sqrt{\sum_{i=1}^d (\overline{x}_i - \underline{x}_i)^2}$.

We now check the three conditions for L_1 consistency of the histogram estimate constructed using `RPQ`.

Proposition 1 (L_1-Consistency)**.** *Let $X_1, X_2, \ldots$ be independent and identical random vectors in $\mathbb{R}^d$ whose common distribution μ has a non-atomic density f, i.e., $\mu \ll \lambda$. Let $\{S_n(i)\}_{i=0}^{\dot{I}}$ on $\mathbb{S}_{0:\infty}$ be the Markov chain formed using `RPQ` of Algorithm 5 with terminal state $\dot{s}$ and histogram estimate $f_{n,\dot{s}}$ over the collection of partitions $\mathcal{L}_n$. As $n \to \infty$, if $\overline{k}_n \to \infty$, $\overline{k}_n/n \to 0$, $\overline{m}_n \geq n/\overline{k}_n$, and $\overline{m}_n/n \to 0$ then the density estimate $f_{n,\dot{s}}$ is strongly consistent in L_1, i.e.*

$$\int |f(x) - f_{n,s}(x)| dx \to 0 \text{ with probability } 1.$$

Proof. We will assume that $\overline{k}_n \to \infty$, $\overline{k}_n/n \to 0$, $\overline{m}_n \geq n/\overline{k}_n$, and $\overline{m}_n/n \to 0$, as $n \to \infty$, and show that the three conditions:

(a) $n^{-1} m(\mathcal{L}_n) \to 0$,
(b) $n^{-1} \log \Pi^*_n(\mathcal{L}_n) \to 0$, and
(c) $\mu(x : \mathrm{diam}(\boldsymbol{x}(x)) > \gamma) \to 0$ with probability 1 for every $\gamma > 0$,

are satisfied. Then by Theorem 1 of Lugosi and Nobel (1996) our density estimate $f_{n,\dot{s}}$ is strongly consistent in L_1.

Condition (a) is satisfied by the assumption that $\overline{m}_n/n \to 0$ since $m(\mathcal{L}_n) \leq \overline{m}_n$ (see Remark 1).

The largest number of distinct partitions of any n point subset of $\mathbb{R}^d$ that are induced by the partitions in $\mathcal{L}_n$ is upper bounded by the size of the collection of partitions $\mathcal{L}_n \subseteq \mathbb{S}_{0:\overline{m}_n-1}$, i.e.

$$\Pi_n^*(\mathcal{L}_n) \leq |\mathcal{L}_n| \leq \sum_{i=0}^{\overline{m}_n-1} C_i$$

where i is the number of splits.

The growth function is thus bounded by the total number of partitions with 0 to $\overline{m}_n - 1$ splits, i.e. the $(\overline{m}_n - 1)$-th partial sum of the Catalan numbers. The partial sum can be asymptotically approximated as (see Appendix A for proof)

$$\sum_{k=0}^{\overline{m}_n-1} C_k \to \frac{4^{\overline{m}_n}}{\left(3(\overline{m}_n - 1)\sqrt{\pi(\overline{m}_n - 1)}\right)} \qquad \text{as } \overline{m}_n \to \infty \quad .$$

Taking logs and dividing by n on both sides we get

$$\begin{aligned}\log \Pi_n^*(L_n)/n &\leq \log\left(\frac{4^{(m(\mathcal{L}_n)+1)}}{3m(\mathcal{L}_n)\sqrt{\pi m(\mathcal{L}_n))}}\right)/n \\ &\leq \tfrac{1}{n}(m(\mathcal{L}_n)+1)\log 4 - \tfrac{1}{n}\log 3\sqrt{(\pi)} - \tfrac{3}{2n}\log m(\mathcal{L}_n).\end{aligned}$$

The first and third term goes to 0 by an application of condition (a). The second term which is just a constant divided by n also vanishes as $n \to \infty$. Therefore, condition (b) is satisfied.

Fix $\gamma, \xi > 0$. There exists a box $\hat{\boldsymbol{x}} = [-M, M]^d$ for a large enough M, such that, $\mu(\hat{\boldsymbol{x}}^c) < \xi$. Consequently,

$$\mu(\{x : \operatorname{diam}(\boldsymbol{x}(x)) > \gamma\}) \leq \xi + \mu(\{x : \operatorname{diam}(\boldsymbol{x}(x)) > \gamma\} \cap \hat{\boldsymbol{x}}).$$

Using 2^{di} hypercubes of equal volume $(2M)^d/2^{di}$, $i = \left\lceil \log_2\left(2M\sqrt{d}/\gamma\right)\right\rceil$ with side length $2M/2^i$ and diameter $\sqrt{d(\frac{2M}{2^i})^2}$, we can have at most 2^{di} boxes in the interior of $\hat{\boldsymbol{x}}$ and δ boxes at the lower dimensional boundaries of $\hat{\boldsymbol{x}}$, i.e. there are at most m_γ disjoint boxes in $\hat{\boldsymbol{x}}$ that have diameter greater than γ, where

$$m_\gamma < 2^{di} + \delta, \quad \delta = \left(2^d + \sum_{j=1}^{d-1} 2^{d-j}\binom{d}{j}2^{ij}\right) \quad . \tag{3.1}$$

By choosing i large enough we can upper bound m_γ by $(2M\sqrt{d}/\gamma)^d + 2^d + \sum_{j=1}^{d-1} 2^{d-j}\binom{d}{j}(2M\sqrt{d}/\gamma)^j$, a quantity that is independent of n, such that

$$\begin{aligned}
\mu(x : \mathrm{diam}(\boldsymbol{x}(x)) > \gamma) &\leq \xi + \mu\left(\{x : \mathrm{diam}(\boldsymbol{x}(x)) > \gamma\} \cap \hat{\boldsymbol{x}}\right) \\
&\leq \xi + m_\gamma \left(\max_{\boldsymbol{x} \in \ell(\dot{s})} \mu(\boldsymbol{x}) \right) \\
&\leq \xi + m_\gamma \left(\max_{\boldsymbol{x} \in \ell(\dot{s})} \mu_n(\boldsymbol{x}) + \max_{\boldsymbol{x} \in \ell(\dot{s})} |\mu(\boldsymbol{x}) - \mu_n(\boldsymbol{x})| \right) \\
&\leq \xi + m_\gamma \left(\frac{\overline{k}_n}{n} + \sup_{\boldsymbol{x} \in \mathbb{R}^d} |\mu(\boldsymbol{x}) - \mu_n(\boldsymbol{x})| \right) .
\end{aligned}$$

The first term in the parenthesis converges to zero since $\overline{k}_n/n \to 0$ by assumption. For $\epsilon > 0$, the second term goes to zero by applying the Vapnik-Chervonenkis theorem to boxes in $\mathbb{R}^d$ with shatter coefficient $s(\mathbb{R}^d, n) = 2^{2d}$ (Devroye et al., 1996, p. 220), i.e.

$$Pr\left\{ \sup_{\boldsymbol{x} \in \mathbb{R}^d} |\mu_n(\boldsymbol{x}) - \mu(\boldsymbol{x})| > \epsilon \right\} \leq 8 \cdot 2^{2d} \cdot e^{-n\epsilon^2/32} \quad .$$

By the Borel-Cantelli lemma,

$$\lim_{n \to \infty} \sup_{\boldsymbol{x} \in \mathbb{R}^d} |\mu_n(\boldsymbol{x}) - \mu(\boldsymbol{x})| = 0 \quad \text{w.p. } 1 \ \ .$$

Thus for any $\gamma, \xi > 0$,

$$\limsup_{n \to \infty} \mu(\{x : \mathrm{diam}(\boldsymbol{x}(x)) > \gamma\}) \leq \xi.$$

Therefore, condition (c) is satisfied and this completes the proof. $\square$

Remark 1. *We can choose $\overline{k}_n$ to be some sub-linear function of n, say n^α. Then $\alpha > 0$ so that $\overline{k}_n \to \infty$ and $\alpha < 1$ so that $\overline{k}_n/n \to 0$. Now let $\overline{m}_n = n^\beta$, then $\beta > 0$ so that $\overline{m}_n \geq n/\overline{k}_n$. The above constraints imply that $\alpha + \beta \geq 1$. Finally $\beta < 1$ such that $\overline{m}_n/n \to 0$.*

3.4 Complexity

We want the subsequent evaluation of $f_{n,s}(x)$ at any point $x \in \mathbb{R}^d$ after the construction, to be computationally efficient in space and time as a function of n and d. Here we focus on the issues involved after an estimator has been constructed. For example, take any kernel density estimate

$$f_{n,h,K}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right) \quad ,$$

where K is a kernel function if $\int |K| < \infty$ and $\int K = 1$. Assume that the optimal h and K for a given n have already been obtained. If it costs $O(d)$ time units to evaluate each

$K(x - X_i/h)$ and $O(d)$ space units to store X_i for each $i = 1, 2, \ldots, n$, then it costs $O(nd)$ time units to evaluate $f_{n,h,K}(x)$ for any x and $O(nd)$ space units to store the n data points in d dimensions that are needed in this evaluation. We consider these $O(nd)$ measures of computational efficiency in space and time for kernel density estimates to provide us with minimum standards in this post-constructive setting. Computational efficiency in the post-constructive setting is indeed important because we typically want to further process the information in f_n to aid subsequent decisions, such as:

- use the density estimate f_n as an approximation for the likelihood of a parameter θ used to simulate the summary statistics $X_1, \ldots, X_n$ in $\mathbb{R}^d$ in a class of approximate Bayesian computations (Fearnhead and Prangle, 2011),

- finding marginal density estimates from f_n along specific coordinate subsets of $\{1, 2, \ldots, d\}$ and highest posterior density regions from f_n which returns $\breve{\ell}(s)$, a subset of $\ell(s)$ (Harlow et al., 2012),

- call other randomised algorithms that take as input k such histogram estimates, say $f_n^{(1)}, \ldots, f_n^{(k)}$ (Devroye and Lugosi, 2001, Chap. 8), and

- arithmetic averaging of $(f_n^{(1)} + \cdots + f_n^{(k)})/k$ (Chapter 4).

We design our asymptotically consistent adaptive histogram estimators $f_{n,\dot{s}}$ given by `RPQ`, as Markov chains over $\mathbb{S}_{0:\infty}$ with a random stopping time, to have no more than $O(nd)$ space requirements. An SRP histogram $f_{n,\dot{s}}$, after it has been constructed, can be thought of as an SRP tree $\dot{s}$ with nodes in $\mathbb{V}(\dot{s})$, such that each $v \in \mathbb{V}(\dot{s})$ is mapped to a box $\boldsymbol{x}_{\rho v} \in \mathbb{IR}^d$ and each leaf node $u \in \mathbb{V}_\ell(\dot{s})$ is further mapped to the 'height' value $f(u)$.

This reduction in memory requirements in the post-constructive phase is due to dropping all the pointers to the data points from the leaf nodes in $\mathbb{V}_\ell(\dot{s})$ and dropping all the recursively computable statistics T at the nodes in $\mathbb{V}(\dot{s})$. Thus, to machine-represent $f_{n,\dot{s}}$, we only need $O(2d)$ units to store the interval vector or box $\boldsymbol{x}_{\rho v}$ associated with each node $v \in \mathbb{V}(\dot{s})$ and $O(1)$ units to store the height value $f(u)$ associated with each leaf node $u \in \mathbb{V}_\ell(\dot{s})$. Since $|\mathbb{V}(\dot{s})| = 2|\mathbb{V}_\ell(\dot{s})| - 1$ for an ordered binary tree such as $\dot{s}$ and the number of leaf nodes in $\dot{s}$ is upper-bounded by $\overline{m}_n$, i.e., $|\mathbb{V}_\ell(\dot{s})| \leq \overline{m}_n$ by Algorithm 5, the post-constructive space requirements for an SRP histogram $f_{n,\dot{s}}$ is bounded by $O(2d(2\overline{m}_n + 1) + \overline{m}_n)$. And clearly this bound is no larger than $O(nd)$, the memory requirement of any kernel density estimate, provided the maximum number of leaves $\overline{m}_n$ satisfies

$$\overline{m}_n < \frac{(n-2)d}{4d+1} \tag{3.2}$$

for each fixed n and d, and the ratio of the number of leaves to the number of data points satisfies

$$\frac{\overline{m}_n}{n} < \frac{1}{n}\frac{(n-2)d}{4d+1} \underset{n\to\infty}{\to} \frac{d}{4d+1} \tag{3.3}$$

as $n \to \infty$. We can easily obtain $\overline{m}_n$, the maximum number of leaves allowed in the SRP histogram estimate $f_{n,\dot{s}}$ from `RPQ` to be an appropriately growing sub-linear function of n that satifies the above inequality to ensure memory requirements of $O(nd)$ (see Remark 2). Thus, by design, the space requirements of $f_{n,\dot{s}}$, upon termination of $\texttt{RPQ}(\psi \circ T)$, grows no faster than the space requirements of any asymptotically consistent kernel density estimator.

Remark 2. *From Remark 1, we chose* $\overline{k}_n$ *to be* $n^{\alpha}, 0 < \alpha < 1$, *and* $\overline{m}_n = n^{\beta}, 0 < \beta < 1, \alpha + \beta \geq 1$. *Replace* $\overline{m}_n$ *with* n^{β} *in Equation 3.2 to obtain*

$$\begin{aligned} n^{\beta} &< \frac{(n-2)d}{4d+1} \ , \\ \beta &< \log\left(\frac{(n-2)d}{4d+1}\right) / \log n < 1 \ . \end{aligned} \tag{3.4}$$

We then use the constraint $\alpha + \beta \geq 1$ *and Equation 3.4 to get memory-justified bounds for* α *and* β, *i.e.*

$$\begin{aligned} 1-\alpha \quad &\leq \quad \beta < \log\left(\frac{(n-2)d}{4d+1}\right) / \log n \ , \\ 1 - \log\left(\frac{(n-2)d}{4d+1}\right) / \log n \quad &< \quad \alpha < 1 \ . \end{aligned}$$

We now study the time requirements for evaluating $f_{n,\dot{s}}(x)$ at any point $x \in \boldsymbol{x}_{\rho} \in \mathbb{R}^d$. We can obtain a pointwise extension $f_{n,\dot{s}}(x) : \boldsymbol{x}_{\rho} \to [0,\infty)$ of SRP histogram tree $f_{n,\dot{s}}$ by associating each point x in the root box $\boldsymbol{x}_{\rho}$ with $f(\rho\iota(x))$, where $x \in \boldsymbol{x}_{\rho\iota(x)} \in \ell(\dot{s})$, i.e., $\rho\iota(x)$ is the node label of the unique leaf box $\boldsymbol{x}_{\rho\iota(x)}$ in $\ell(\dot{s})$ that contains x. Due to the recursive partitioning structure of $\dot{s}$, we need at most $2d$ inequality checks through at most $O(\log_2(\overline{m}_n))$ many internal nodes of $\dot{s}$ to find $\boldsymbol{x}_{\rho\iota(x)}$ for a given $x \in \boldsymbol{x}$ and then simply look-up its height $f(\boldsymbol{x}_{\rho\iota(x)})$, as opposed to the constant $O(nd)$ kernel-specific operations required for evaluating $f_{n,h,K}(x)$ at any $x \in \boldsymbol{x}_{\rho}$ for any kernel density estimate.

Chapter 4

Posterior Expectation of SRP Histograms

This is joint work with Dr. Raazesh Sainudiin, Jennifer Harlow and Dr. Dominic Lee from the Department of Mathematics and Statistics, University of Canterbury. The journal article version of this chapter has been accepted for publication in the ACM Transactions on Modeling and Computer Simulation (Special Issue on Monte Carlo Methods in Statistics).

4.1 Posterior Mean of SRP Histograms

Recall that the SRP histogram estimate is defined as follows:

$$f_{n,s}(x) = \begin{cases} \frac{\mu_n(\boldsymbol{x}(x))}{\mathrm{vol}\,(\boldsymbol{x}(x))} = \frac{\#\boldsymbol{x}(x)}{n} \times \frac{1}{\mathrm{vol}\,(\boldsymbol{x}(x))} & \text{if } 0 < \mathrm{vol}\,(\boldsymbol{x}(x)) < \infty, \boldsymbol{x}(x) \in \ell(s), \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

Let the height-value of a given node ρv of an SRP histogram be given by $h(\rho v) = \#\boldsymbol{x}_{\rho v}/(n \cdot \mathrm{vol}\,(\boldsymbol{x}_{\rho v}))$. Using our SRP information structures, we are interested in producing the sample mean histogram estimate of the Bayes posterior expectation over this space of SRP histograms.

Let π be the posterior distribution that is proportional to the product of the likelihood of the data given s and the prior probability of s, i.e.

$$\begin{aligned} \pi(s) &:= \Pr\{s|x_1,\ldots,x_n\} \\ &\propto \Pr\{x_1,\ldots,x_n|s\}\Pr\{s\} \\ &= \Pr\{x_1|s\}\Pr\{x_2|s\}\cdots\Pr\{x_{n-1}|s\}\Pr\{x_n|s\}\Pr\{s\} \\ &\approx \prod_{i=1}^{n} f_{n,s}(x_i)\Pr\{s\} \\ &= \prod_{\boldsymbol{x}_{\rho v}\in\ell(s)} \left(\frac{\#\boldsymbol{x}_{\rho v}}{n \cdot \mathrm{vol}\,(\boldsymbol{x}_{\rho v})}\right)^{\#\boldsymbol{x}_{\rho v}} \Pr\{s\} \ . \end{aligned}$$

Note how we approximate the likelihood of the data given s by the maximum likelihood value from the histogram on s.

We want our prior distribution $\{\Pr(s)\}$ over $s \in \mathbb{S}_{0:\infty}$ to be proper and uninformative in some natural sense. Moreover, we also want our prior probabilities to decrease as the partition size increases in order to penalize large partitions. With these considerations, we propose a Catalan family of proper priors associated with any convergent decreasing sequence. Suppose $\{a_k\}$ for $k = 1, 2, \ldots$ is any decreasing sequence of positive real numbers such that $\sum_{k=1}^{\infty} a_k = a < \infty$. Recall from Equation (2.1) that the Catalan number C_k gives the number of SRPs in $\mathbb{S}_k$ with k splits and $k+1$ leaves. An $\{a_k\}$-*penalized uninformative proper Catalan prior* that assigns states in $\mathbb{S}_k$ with probability a_k/a and distributes this mass uniformly over $\mathbb{S}_k$ is given by

$$\Pr\{s\} = \sum_{k=0}^{\infty} \mathbb{1}_{\mathbb{S}_k}(s) \frac{a_k}{aC_k} \ . \tag{4.2}$$

In this work we fix a particular prior obtained from the sequence of $a_k = 1/C_k$ with $a = 2 + 4\pi/3^{5/2} \approx 2.806133050770763$ (McGarvey and Cloitre, 2005). Such a *natural Catalan prior* is given by

$$\Pr\{s\} = \sum_{k=0}^{\infty} \mathbb{1}_{\mathbb{S}_k}(s) \frac{1}{(2 + 4\pi/3^{5/2})C_k^2} \ . \tag{4.3}$$

Thus, the posterior distribution on $\mathbb{S}_{0:\infty}$ is given by

$$\pi(s) \propto \prod_{\boldsymbol{x}_{\rho v} \in \ell(s)} \left(\frac{\#\boldsymbol{x}_{\rho v}}{n \cdot \mathrm{vol}\,(\boldsymbol{x}_{\rho v})} \right)^{\#\boldsymbol{x}_{\rho v}} \cdot \sum_{k=0}^{\infty} \mathbb{1}_{\mathbb{S}_k}(s) \frac{1}{(2 + 4\pi/3^{5/2})C_k^2} \ . \tag{4.4}$$

We can obtain an estimate of the posterior mean from the sample average of the thinned-out post-burnin sequence of states visited by a discrete time Markov chain $\{S(t)\}$, for $t \in \mathbb{Z}_+ := \{0, 1, 2, \ldots\}$, over the state space $\mathbb{S}_{0:\infty}$ with the posterior distribution π as its stationary distribution. In order to do this we need to be able to efficiently obtain the average of a number of SRP histograms.

4.2 Averaging SRP Histograms

We can average m histograms if we are able to add any two histograms together and get another histogram on the condition that each histogram has the same root box, and multiply any histogram by a scalar. Since the RP trees are closed under pair-wise union or overlay operations, we can extend these operations to SRPs from which the histograms are built. This enables us to perform arithmetic over SRPs in a recursive and efficient manner to obtain averaged histograms.

Consider two SRPs $s^{(1)}$ and $s^{(2)}$ with root nodes $\rho^{(1)}$ and $\rho^{(2)}$, respectively, with the same root box $\boldsymbol{x}_\rho = \boldsymbol{x}_{\rho^{(1)}} = \boldsymbol{x}_{\rho^{(2)}}$. Let the corresponding histograms of SRPs $s^{(1)}$ and $s^{(2)}$ be $f_{n,s^{(1)}}$ and $f_{n,s^{(2)}}$. We can add the two histograms by applying `AddSRPHist`($\rho^{(1)}$, $\rho^{(2)}$) described by Algorithm 6.

Algorithm 6: `AddSRPHist`

input : two SRP histogram root nodes $\rho^{(1)}$ and $\rho^{(2)}$ with same root box $\boldsymbol{x}_\rho = \boldsymbol{x}_{\rho^{(1)}} = \boldsymbol{x}_{\rho^{(2)}}$.

output : the sum SRP histogram with root node $\rho^{(1)+(2)}$.

Make a new node $\rho^{(1)+(2)}$ with box $\boldsymbol{x}_\rho$

$h(\rho^{(1)+(2)}) \leftarrow h(\rho^{(1)}) + h(\rho^{(2)})$

if ($\rho^{(1)}$ *is a leaf node*) & ($\rho^{(2)}$ *is not a leaf node*) **then**

  Make nodes L', R'

  $\boldsymbol{x}_{\mathsf{L}'} \leftarrow \boldsymbol{x}_{\rho^{(1)}\mathsf{L}}$, $\boldsymbol{x}_{\mathsf{R}'} \leftarrow \boldsymbol{x}_{\rho^{(1)}\mathsf{R}}$

  $h(\mathsf{L}') \leftarrow h(\rho^{(1)})$, $h(\mathsf{R}') \leftarrow h(\rho^{(1)})$

  Graft onto $\rho^{(1)+(2)}$ as left child the node `AddSRPHist`$(\rho^{(2)}\mathsf{L}, \mathsf{L}')$

  Graft onto $\rho^{(1)+(2)}$ as right child the node `AddSRPHist`$(\rho^{(2)}\mathsf{R}, \mathsf{R}')$

end

if ($\rho^{(2)}$ *is a leaf node*) & ($\rho^{(1)}$ *is not a leaf node*) **then**

  Make nodes L', R'

  $\boldsymbol{x}_{\mathsf{L}'} \leftarrow \boldsymbol{x}_{\rho^{(2)}\mathsf{L}}$, $\boldsymbol{x}_{\mathsf{R}'} \leftarrow \boldsymbol{x}_{\rho^{(2)}\mathsf{R}}$

  $h(\mathsf{L}') \leftarrow h(\rho^{(2)})$, $h(\mathsf{R}') \leftarrow h(\rho^{(2)})$

  Graft onto $\rho^{(1)+(2)}$ as left child the node `AddSRPHist`$(\rho^{(1)}\mathsf{L}, \mathsf{L}')$

  Graft onto $\rho^{(1)+(2)}$ as right child the node `AddSRPHist`$(\rho^{(1)}\mathsf{R}, \mathsf{R}')$

end

if (*both* $\rho^{(1)}$ *and* $\rho^{(2)}$ *are not leaf nodes*) **then**

  Graft onto $\rho^{(1)+(2)}$ as left child the node `AddSRPHist`$(\rho^{(1)}\mathsf{L}, \rho^{(2)}\mathsf{L})$

  Graft onto $\rho^{(1)+(2)}$ as right child the node `AddSRPHist`$(\rho^{(1)}\mathsf{R}, \rho^{(2)}\mathsf{R})$

end

return $\rho^{(1)+(2)}$

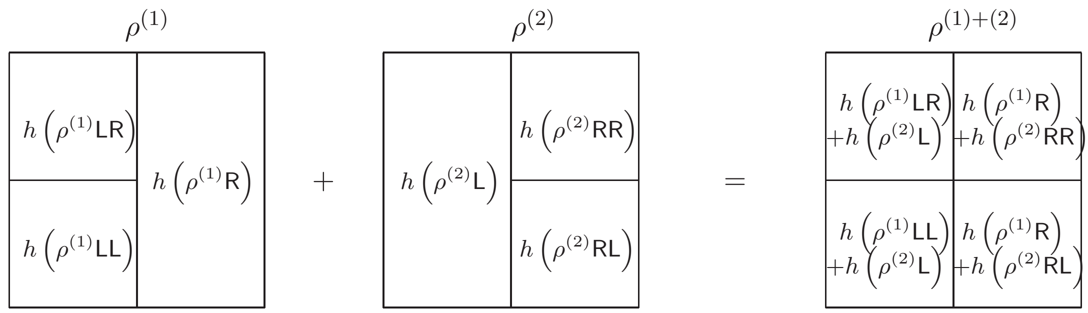


Figure 4.1: Adding two SRP histograms.

Figure 4.1 illustrates how addition is performed on two SRPs to obtain the sum SRP. The sum is then divided by 2 to obtain the average histogram $f_{n,s^{(1)+(2)}} = (f_{n,s^{(1)}} + f_{n,s^{(2)}})/2$. The average of m SRPs is obtained similarly as $((((f_{n,s^{(1)}} + f_{n,s^{(2)}}) + f_{n,s^{(3)}}) + \cdots) + f_{n,s^{(m)}})/m$.

4.3 A Metropolis-Hastings Markov Chain Over SRP Histograms

In this section, we will show that for a given set of n data points $x_1, x_2, \ldots, x_n$ in the root box $\boldsymbol{x}_\rho$, the base chain $\{Y(t)\}_{t\in\mathbb{Z}_+}$ is irreducible and aperiodic on the machine-representable finite state space $\tilde{\mathbb{S}}_n \subset \mathbb{S}_{0:\infty}$ and then derive a Metropolis-Hastings chain $\{S(t)\}_{t\in\mathbb{Z}_+}$ with the desired stationary distribution. We conclude the section with some heuristics to diagnose and accelerate mixing.

4.3.1 Machine-representable State Space

Let us note that on a computer with a floating-point number screen of fixed precision, such as `double` precision numbers, $\tilde{\mathbb{S}}_n$ is necessarily finite subset of $\mathbb{S}_{0:\infty}$. For instance, we cannot represent the child boxes with our number screen if the widest side of the parent box to be bisected is already given by two adjacent floating-point numbers. First, let us appreciate how the data-dependent states in $\tilde{\mathbb{S}}_n$ are dictated by the boundary of "unsplittable states" in $\mathbb{S}_{0:\infty}$ in addition to the hard boundaries imposed by the machine's number screen. There are three natural ways to define the boundary of unsplittable SRP states.

The first and simplest way to define such a boundary in $\mathbb{S}_{0:\infty}$ is by saying that we cannot split an SRP state beyond a maximal number of splits $\overline{m}_n$. We can allow this $\overline{m}_n = n^\beta$ for some $\beta < 1$ in order to ensure a sub-linear growth of the number of leaves with the number of data points, i.e. $\overline{m}_n/n \to 0$ as $n \to \infty$, for instance. Using this hard boundary based on the maximum number of allowed splits our finite state space of the base chain is $\tilde{\mathbb{S}}_n = \mathbb{S}_{0:\overline{m}_n}$. In this case, we know that $|\mathbb{S}_{0:\overline{m}_n}| = \sum_{k=0}^{\overline{m}_n} C_k$. We typically take β close to 1 under this approach to ensure a sufficiently large $\tilde{\mathbb{S}}_n$.

The second and less simple way to define such a boundary in $\mathbb{S}_{0:\infty}$ is by saying that we cannot split any leaf of an SRP state beyond a minimum volume, say $\underline{\lambda}_n$. Recall that the volume of a box at a leaf node ρv at depth δ in an SRP s is $\mathrm{vol}\,(\boldsymbol{x}_{\rho v}) = \mathrm{vol}\,(\boldsymbol{x}_\rho)/2^\delta$. Thus the minimum volume boundary constraint is equivalent to a maximal leaf-depth constraint on states in $\tilde{\mathbb{S}}_n$, where we ensure that all states visited by the chain have an $\underline{\lambda}_n$-dependent maximal depth $\overline{\delta}_n = \lfloor \log_2 (\mathrm{vol}\,(\boldsymbol{x}_\rho)/\underline{\lambda}_n) \rfloor$. This gives an upper bound by considering the tip of the parabolic tongue of the unsplittable boundary of leaf-depth constrained SRP states in $\mathbb{S}_{0:\infty}$ over an arrangement similar to the transition diagram of Figure 2.2. In this case, $\tilde{\mathbb{S}}_n$ is contained in $\mathbb{S}_{0:\tau}$, where $\tau = 2^{\overline{\delta}_n} - 1$. Once again we can allow the maximal depth $\overline{\delta}_n$ to increase appropriately with the number of data points n to ensure that $\underline{\lambda}_n \to 0$ as $n \to \infty$.

The third, more complicated, and more data-driven way to define such a boundary of unsplittable states in $\mathbb{S}_{0:\infty}$ in order to produce a finite data-dependent state space $\tilde{\mathbb{S}}_n$ for our base chain is by looking at the number of points inside the boxes. Here we allow only a leaf box to be split if each resulting child box will have at least $\#_n$ points in it except when one of the child boxes will be empty and the other child box will get all of the $\#_n$ or more points from its parent box. This way we ensure that all the leaves of a tree in $\tilde{\mathbb{S}}_n$ have at least $\#_n$ points, provided they have any points at all. This splitting procedure is more data-driven than the previous two and also concentrates the base chain over SRP states whose partitions refine on the locations of the given data set $x_1, x_2, \ldots, x_n$. Observe that for a tree s at the boundary of splittable states in $\tilde{\mathbb{S}}_n$ only a subset of its leaf nodes are splittable by this criterion. For instance, a leaf node with no data points in its associated leaf box cannot be split further and a leaf box with at least $\#_n$ many points in its leaf box cannot be split further if the split will result in a non-empty child box with fewer that $\#_n$ points in it. We refer to the set of leaf boxes corresponding to this set of splittable leaf nodes of s as $\hat{\ell}(s)$ and note that $\hat{\ell}(s) \subseteq \ell(s)$ for all $s \in \tilde{\mathbb{S}}_n$. We take $\#_n$ to either be a constant, say 1 or 2, for all n and thereby ensure that $\tilde{\mathbb{S}}_n$ is one of the largest finite state spaces, or we allow $\#_n$ to be a sub-linear function of n, say $\#_n = n^\alpha$ with an appropriate $\alpha < 1$, in order to parametrically control the size of $\tilde{\mathbb{S}}_n$. In all the simulations carried out here, we use this $\#_n$-specified splitting rule and fix $\#_n = 1$ in order to conservatively work with a large state space $\tilde{\mathbb{S}}_n$. The finite state space $\tilde{\mathbb{S}}_n$ determined by such a $\#_n$-specified splitting rule has advantages due to its data-dependent partitioning nature and thereby better computational performance (space and time requirements) when compared to the exclusively n-dependent global splitting rules based on $\overline{m}_n$, the maximal number of splits allowed or on $\overline{\delta}_n$, the maximal depth of a leaf node. We sometimes use these two globally determined boundary of unsplittable states in logical conjunction with this $\#_n$-based data-dependent rule in order to study the effects of $\overline{\delta}_n$ and $\overline{m}_n$ on the jointly determined state space $\tilde{\mathbb{S}}_n$. In all simulation experiments of Chapter 5 we found that the effect of the $\#_n$-determined state space $\tilde{\mathbb{S}}_n$ is minimal on the performance of the posterior mean histogram estimate, provided $\tilde{\mathbb{S}}_n$ is large enough to contain partitions that can represent the underlying unknown density. We observe no significant change in integrated absolute errors when $\#_n = 0, 1, 2, 3$ because states close to these boundaries have the most number of leaves with either 0 or $\#_n$ points in their leaf boxes and such states are rarely visited by our Markov chain

4.3.2 Base Markov Chain

Consider the *stay-split-merge* base Markov chain $\{Y(t)\}_{t\in\mathbb{Z}_+}$ on the state space $\tilde{\mathbb{S}}_n$ with initial state $Y(0) = s_0$. We propose to stay in the current state with positive probability σ and move to another state with probability $1-\sigma$. If a move is chosen, it can be a permitted bisection or a reunion with equally probability $(1-\sigma)/2$. If a bisection is chosen, each splittable leaf node in the current state s has an equal probability $(1-\sigma)/2|\hat{\ell}(s)|$ of being bisected. Similarly, if a reunion is chosen, each cherry node in s has an equal probability $(1-\sigma)/2|c(s)|$ of having its sibling nodes reunited to itself. Then, the transition probabilities between any two states $s, s' \in \tilde{\mathbb{S}}_n$ are

$$Q(s,s') = \begin{cases} (1-\sigma)/2|\hat{\ell}(s)| & \text{if } s \text{ can be split once to get } s' \\ (1-\sigma)/2|c(s)| & \text{if } s \text{ can be reunited once to get } s' \\ \sigma & \text{if } s = s' \\ 0 & \text{otherwise} \end{cases} . \tag{4.5}$$

The chain $\{Y(t)\}_{t\in\mathbb{Z}_+}$ on the finite state space $\tilde{\mathbb{S}}_n$, that is obtained by the $\#_n$-specified splitting rule with $\#_n = 1$ for instance, is irreducible since we can eventually go from any state s to any other state s' and vice versa by reuniting cherries to reach s_0 from s and then from s_0 to s' by selectively splitting leaves (in effect by reversing the reunion operations that take you from s' to s_0). Note that any SRP state can be reached from the root SRP s_0 by a sequence of selective splits. The chain is also aperiodic since there is a positive probability σ of staying in the current state. Therefore, the base chain has a unique stationary distribution.

4.3.3 Metropolis-Hasting Markov Chain

Using the irreducible and aperiodic base chain $\{Y(t)\}_{t\in\mathbb{Z}_+}$ on the finite state space $\tilde{\mathbb{S}}_n$ with transition matrix Q in Equation 4.5 and the posterior distribution π given in Equation 4.4, we can now proceed to construct a Metropolis-Hastings chain $\{S(t)\}_{t\in\mathbb{Z}_+}$ on $\tilde{\mathbb{S}}_n$ with π truncated to $\tilde{\mathbb{S}}_n$ as its stationary distribution with the following transition probabilities

$$P(s,s') = \begin{cases} Q(s,s')a(s,s') & \text{if } s \text{ leads to } s' \text{ by a split or a merge} \\ 1 - \sum_{s \in \{z \in \mathbb{S}: z \neq s\}} Q(s,z)a(s,z) & \text{if } s = s' \\ 0 & \text{otherwise} \end{cases} ,$$

where the acceptance probability is given by

$$a(s,s') := \min\left\{1, \frac{\pi(s')Q(s',s)}{\pi(s)Q(s,s')}\right\} .$$

4.3.4 Gelman-Rubin Diagnostics

We finally use the heuristic Gelman-Rubin convergence diagnostic statistic (Gelman and Rubin, 1992) to automatically stop $\{S(t)\}$ after producing the desired number of post-burnin thinned-out samples from the desired stationary distribution π normalized over $\tilde{\mathbb{S}}_n$. To obtain this heuristic auto-stopping rule we run C parallel independent Metropolis-Hastings Markov chains $(\{S_1(t)\}, \ldots, \{S_C(t)\})$ for $t \in \{1, 2, \ldots, M\}$ of maximum allowed length M and calculate $\hat{R}(t)$, the ratio of between-sequence variance to the within-sequence variance for a scalar summary of the histogram states in the C chains up to time t. The required samples are obtained by ensuring that $\hat{R}(t)$ remains bounded above by $1 + \epsilon$ throughout the required length of the post-burnin period, where ϵ is typically taken to be 0.1. Finally, the algorithm stops possibly without the needed samples if all C chains have run for the maximum length of M. We focus on the scalar summary of the SRP state that gives the number of leaves.

4.3.5 Initial Condition

If the Metropolis-Hastings chain $\{S(t)\}$ is initialised far from the states with high posterior mass then the mixing time can be prohibitively large. Choosing an initial state $S(0) = \check{s}$ with too few splits or too many splits can lead to poor mixing of $\{S(t)\}$. Thus, we want a heuristic strategy to produce good initial states for $\{S(t)\}$. Algorithm 7 is a simple modification of `RPQ` which traverses through $\tilde{\mathbb{S}}_n$ and returns the state with the maximum log-posterior among all the states visited by `InitSEBPQ` to initialise our Metropolis-Hastings chain $S(t)$. This initialisation strategy, coupled with monitoring the log-posterior and the number of leaves of $S(t)$, leads to reasonable estimates with small integrated absolute errors in all the simulation experiments of Chapter 5.

Algorithm 7: `InitSEBPQ`

input : (i) data $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$;
(ii) s `// initial SRP`

output : $\check{s}$ to initialise the M-H MC $\{S(t)\}$.

initialise: $\mathfrak{S} \leftarrow \emptyset$ `// a list to track SRP histograms`

repeat
- $\boldsymbol{x}_{v^*} \leftarrow$ `Uniform`$(\mathrm{argmax}_{\boldsymbol{x}_{\rho\mathsf{v}} \in \ell(\mathsf{s})} \#\boldsymbol{x}_{\rho\mathsf{v}})$ `// sample a leaf box with most data`
- bisect $\boldsymbol{x}_{v^*}$ of s
- update counts in the child nodes
- $\mathfrak{S}$.append(s) `// update list of states visited`

until $s \in \tilde{\mathbb{S}}_n$;

return $\check{s} \leftarrow \mathrm{argmax}_{s \in \mathfrak{S}} \log(\mathrm{posterior}(\mathrm{f_{n,s}}))$

Chapter 5

Simulation Results from Density Estimation

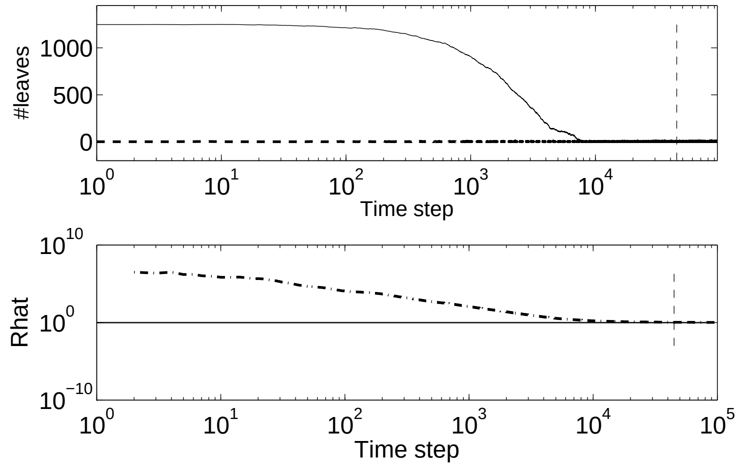
We now present the mean integrated absolute error (MIAE) and standard error (std. err.) for various sample sizes, densities and dimensions to empirically evaluate the performance of our estimator. All of our programs were run on a machine with dual Intel X5670 2.93Ghz 6 core Xeon CPUs, 48GB of RAM, 2 x 320GB 15K SAS hard drives and OpenSuSE 11.2 (x86 64) OS. The three types of densities we estimate are given in the next two sections.

5.1 Multivariate Uniform Densities

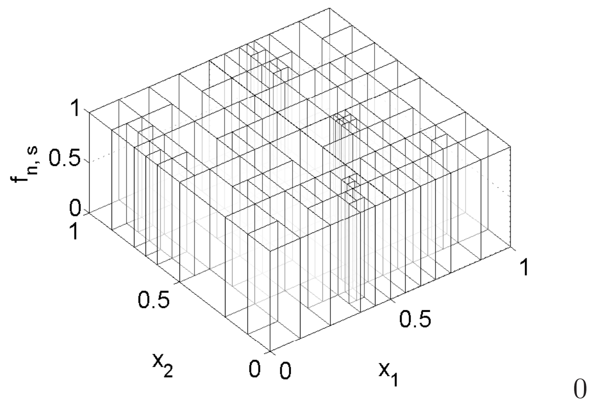
We first test the performance of our density estimator with data simulated from dD-uniform density, i.e., the uniform density on $[0, 1]^d$. For our simulations summarized in Table 5.1, we worked with the uniform density in dimensions 1D, 2D, 10D, 100D, and 1000D for various sample sizes as shown in Table 5.1. A dash (-) is used in Table 5.1 to indicate machine memory (RAM) limitations for the storage of the required data. Mean integrated absolute errors and their standard errors were obtained from 25 replicate data sets for each density. We obtained the posterior mean histogram estimate by averaging over 1000 samples from our Metropolis-Hastings Markov chain $\{S(t)\}$ after burnin with a thin-out rate of 50 samples.

We use the heuristic Gelman-Rubin diagnostics as described in Section 4.3.4 to assess chain convergence and obtain the desired samples for averaging. The two chains have initial states as the root box (only one leaf node) and a state with a large number of leaf nodes respectively, so that we have two starting states that are “far” from each other. Histogram samples from $\{S(t)\}$ are thinned out and collected for averaging after $\hat{R}(t)$ gets below 1.1 Figure 5.1(b) shows the convergence diagnostic plots for the 2D-uniform density. The trace plots for the number of leaves show that the chains eventually hover around states with similar number of leaves (between 1-10 leaves). The posterior mean histogram estimate is given by Figure 5.1(a).

From Table 5.1 it is clear that as the sample size n increases from 10^2 to 10^8 in multiples of 10 the MIAE decreases by at least a factor of 2 across all dimensions $d \in \{1, 2, 10, 100, 1000\}$. Observe how the MIAE of the density estimate of the uniform random vector on the hypercube $[0, 1]^d$ is independent of the dimension d. This is because the target density being estimated for any d, i.e. the uniform density on $[0, 1]^d$ is also the SRP histogram $f_{n,s_0}(x) = \mathbb{1}_{[0,1]^d}(x)$



(a) Trace plots of the number of leaves for two chains and its $\hat{R}$ statistic.



(b) Posterior mean histogram estimate with 55 leaf nodes and IAE 0.0010.

Figure 5.1: A typical run of two Metropolis-Hastings chains based on 10^6 data points from the 2D-uniform density. The chains were started from the root node and a state with 1247 leaf nodes. 1000 histogram samples were collected with a thin out of 50 after burnin at time step 44861.

at the root SRP $s_0 \in \tilde{S}_n$. Observe that our posterior mean estimate is so close in MIAE to the true density and the chains initialised far from s_0 quickly converge to it due to the penalizing effect of our Catalan prior on states with unnecessarily more leaves. We are not aware of other nonparametric multivariate density estimators that can handle the sample sizes and dimensions in Table 5.1 for such highly unstructured data from this family of uniform densities on $[0,1]^d$. On the other hand, as the two families of structured densities in the next section show, our estimator is not immune to the curse of dimensionality when there is underlying structure in the data.

n	1D	2D	10D	100D	1000D
10^2	0.1014 (0.0655)	0.1006 (0.0659)	0.1225 (0.0670)	0.1408 (0.0711)	0.1187 (0.0771)
10^3	0.0380 (0.0231)	0.0333 (0.0221)	0.0294 (0.0178)	0.0330 (0.0204)	0.0386 (0.0231)
10^4	0.0118 (0.0066)	0.0121 (0.0090)	0.0123 (0.0067)	0.0115 (0.0061)	0.0121 (0.0075)
10^5	0.0035 (0.0020)	0.0040 (0.0025)	0.0038 (0.0023)	0.0042 (0.0025)	0.0034 (0.0023)
10^6	0.0011 (0.0006)	0.0012 (0.0006)	0.0013 (0.0006)	0.0011 (0.0007)	0.0012 (0.0010)
10^7	0.0004 (0.0002)	0.0004 (0.0002)	0.0003 (0.0002)	-	-
10^8	0.0001 (0.0001)	0.0001 (0.0001)	-	-	-

Table 5.1: MIAE (std. err.) for n samples from the uniform density in various dimensions. 1000 samples collected. Thin out rate = 50.

5.2 Multivariate Gaussian and Rosenbrock Densities

5.2.1 Approximated Functions

We now test the performance of our density estimator with data simulated from multivariate Gaussian and Rosenbrock densities. The Rosenbrock density in d dimensions over some box $\boldsymbol{x} \in \mathbb{R}^d$ is obtained by appropriately normalizing the Rosenbrock shape given by:

$$r_d(x) = \exp\left(-\sum_{j=2}^{d}(100(x_j - x_{j-1}^2)^2 + (1 - x_{j-1})^2)\right) . \tag{5.1}$$

In our simulation studies, the standard 1D-, 2D-, and 5D-Gaussian densities and the 2D- and 5D-Rosenbrock densities are approximated by simple functions in order to simplify the multivariate integrations during absolute error evaluations, especially in higher dimensions.

The multivariate Gaussian and Rosenbrock densities are approximated using simple functions over SRP partitions and simulated from corresponding mixtures of uniform densities using interval analytic methods of Sainudiin and York (2005). Briefly, using interval arithmetic (Moore, 1967) the range of the target density f over each leaf box $\boldsymbol{x}_{\rho v}$ in $\ell(s)$, a partition of the domain based on the leaves of an RP tree s, is enclosed rigorously in an interval $\boldsymbol{y}_{\rho v}$, i.e., $\boldsymbol{y}_{\rho v} := [\underline{y}_{\rho v}, \overline{y}_{\rho v}] \supseteq \{f(x) : x \in \boldsymbol{x}_{\rho v}\}$. Each leaf box $\boldsymbol{x}_{\rho v}$ is prioritized for the next bisection on the basis of $\mathrm{vol}\,(\boldsymbol{x}_{\rho v}) \times (\overline{y}_{\rho v} - \underline{y}_{\rho v})$, i.e., the uncertainty in the enclosure of $\mu(\boldsymbol{x}_{\rho v}) = \int_{\boldsymbol{x}_{\rho v}} f d\lambda$. The bisection stops once the partition has exactly Λ leaves and the target density at the mid-point of each leaf box is used to construct a simple function approximation to the target density. Finally, this simple function is normalized to give the weighted mixture of uniform densities over the RP tree with Λ leaf boxes. We can easily produce perfect samples to simulate data from this normalized simple function approximation of the target density. The fundamental theorems of interval analysis guarantee that the simple function converges uniformly to the target density as the mesh approaches 0 and $\Lambda = |\ell(s)|$ approaches ∞, provided the target density is given by a locally Lipschitz arithmetical expression (Neumaier, 1990, 2.1.1-3). The Λ in Table 5.2 denotes the number of leaf nodes used to approximate the densities. These simple approximating densities were chosen for two fundamental reasons: (i) to keep the true density that the data is simulated from to lie within the class of SRP histograms for easier interpretation of our simulation results and (ii) to compute the exact IAE by taking advantage of SRP histogram arithmetic (for details see Harlow et al. (2012)).

5.2.2 Estimated Hellinger Distances

We use an estimated Hellinger distance between the original target density f and its Λ-specific approximation f_Λ to quantify the extent of these appproximations. The closer the value is to 1, the closer the estimated density is to the actual density in the Hellinger distance. The Hellinger distance is estimated using the first two sample moments of each density as described in the following.

Let $\hat{m}$ and $\hat{\Sigma}$ be the first two sample moments of the original target density, and let $\hat{m}_\Lambda$ and $\hat{\Sigma}_\Lambda$ be the first two sample moments of the Λ-specific approximation. The computation of the Hellinger distance requires the Bhattacharyya coefficient $BC(f, f_\Lambda) = -\exp\left(D_B(f, f_\Lambda)\right)$, where the value D_B is the Bhattacharya distance between the two densities and is given by

$$D_B = \frac{1}{8}(\hat{m} - \hat{m}_\Lambda)^T \Sigma^{-1} (\hat{m} - \hat{m}_\Lambda) + \frac{1}{2} \ln \left(\frac{\det \Sigma}{\sqrt{\det \hat{\Sigma} \det \hat{\Sigma}_\Lambda}} \right)$$

with $\Sigma = (\hat{\Sigma} + \hat{\Sigma}_\Lambda)/2$. Finally, the Hellinger distance between the two densities is $\sqrt{1 - BC}$.

Figure 5.2 shows the estimated Hellinger distances for various Gaussian (solid lines) and Rosenbrock (dashed lines) densities from their Λ-specific approximations based on 10^7 samples. As the dimension increases, the Hellinger distances increases for each density as well,

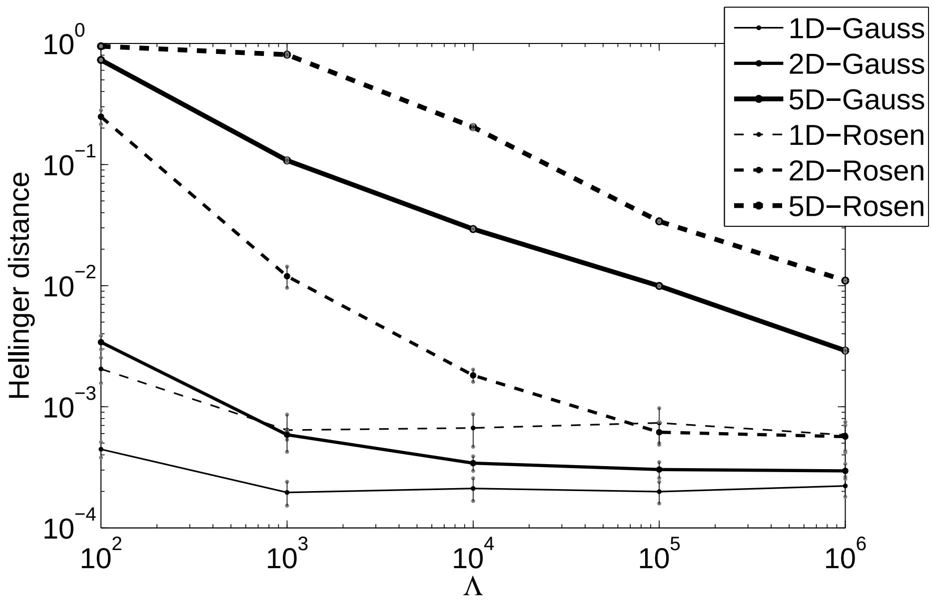
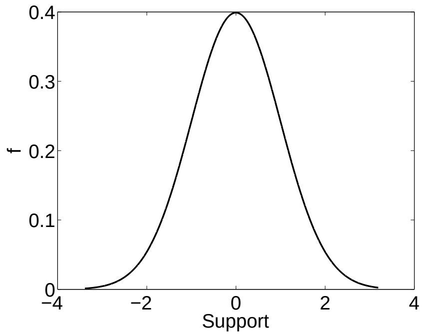


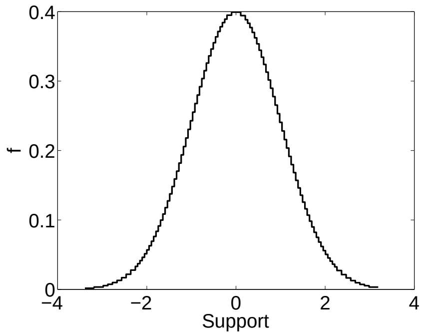
Figure 5.2: Hellinger distances for the Gaussian (solid line) and Rosenbrock (dashed lines) densities.

but decreases and eventually stabilizes as Λ increases. Overall the Hellinger distances for the more complex Rosenbrock densities are higher than that of the centrally concentrated Gaussian densities across dimensions. The curse of dimensionality for non-uniform densities thus manifests itself in terms of the size of Λ needed to approximate it. The estimated Hellinger distances here highlight the limitations of using Λ-specific SRP partitions to approximate a desired density, espeically as the dimension increases. However, the simulation of data from the approximations allows us to compute the integrated absolute errors exactly in high dimensions using arithmetic over trees representing SRP histograms.

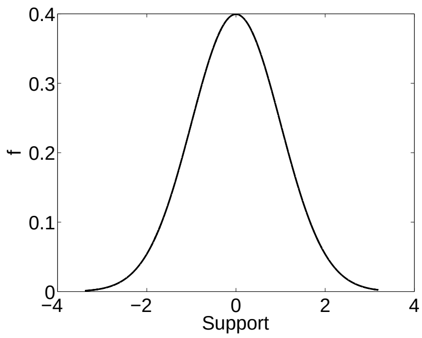
We now look at some Λ-approximated functions for the 1D-Gaussian, 2D-Gaussian and 2D-Rosenbrock densities. The estimated Hellinger distances shown in Figure 5.2 indicate that approximations using $\Lambda = 10000$ and $\Lambda = 1000000$ are reasonable for the 1D- and 2D-Gaussian densities, as featured in the density plots of the 1D-Gaussian (Figure 5.3) and the contour plots of the 2D-Gaussian (Figure 5.4). Both showed that the approximations for $\Lambda = 10000$ and $\Lambda = 1000000$ are closer to the actual density compared to the approximations using $\Lambda = 100$. The 2D-Rosenbrock density can be obtained by using an appropriate constant for the 2D-Rosenbrock shape given in Equation 5.1. Figures 5.5(d) and 5.5(e) show that $\Lambda = 10000$ and $\Lambda = 1000000$ are better approximations compared to $\Lambda = 100$ (Figure 5.5(c)).



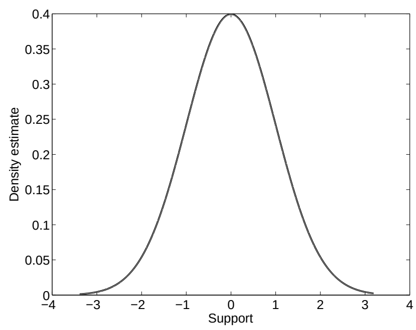
(a) Actual density.



(b) $\Lambda = 100$.

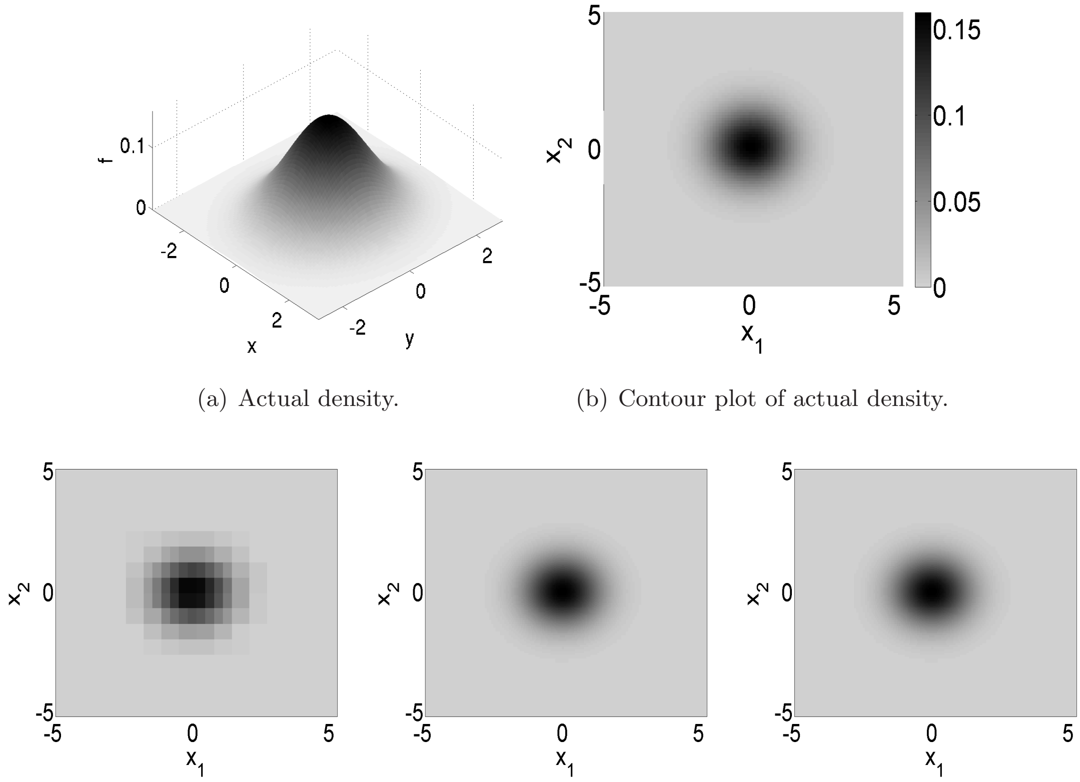


(c) $\Lambda = 10000$.



(d) $\Lambda = 1000000$.

Figure 5.3: Approximated 1D-Gaussian density using $\Lambda = 100, 10000, 1000000$ leaf nodes against the actual 1D-Gaussian density.



(a) Actual density.

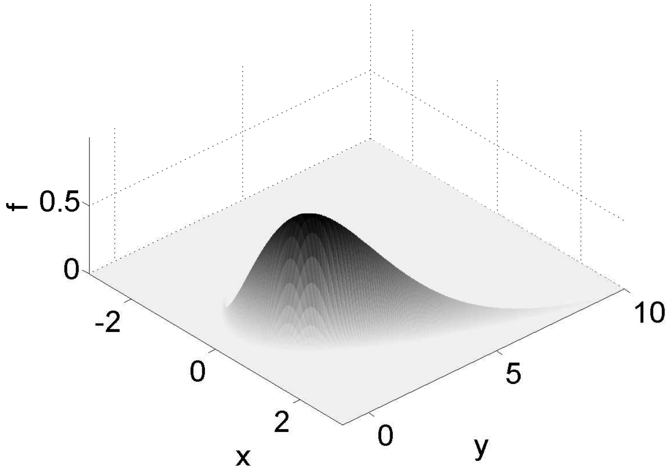
(b) Contour plot of actual density.

(c) $\Lambda = 100$.

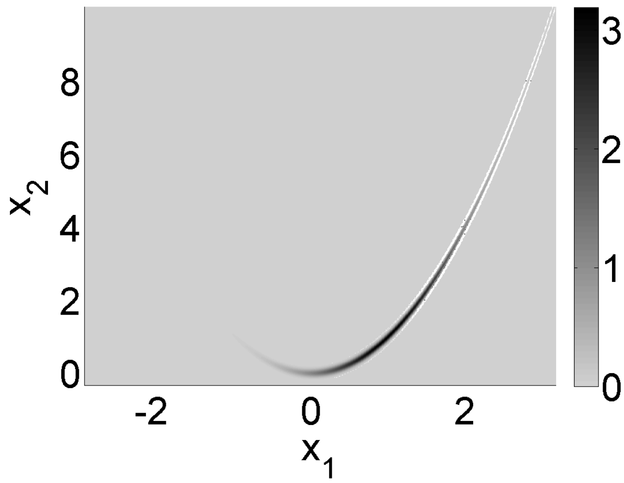
(d) $\Lambda = 10000$.

(e) $\Lambda = 1000000$.

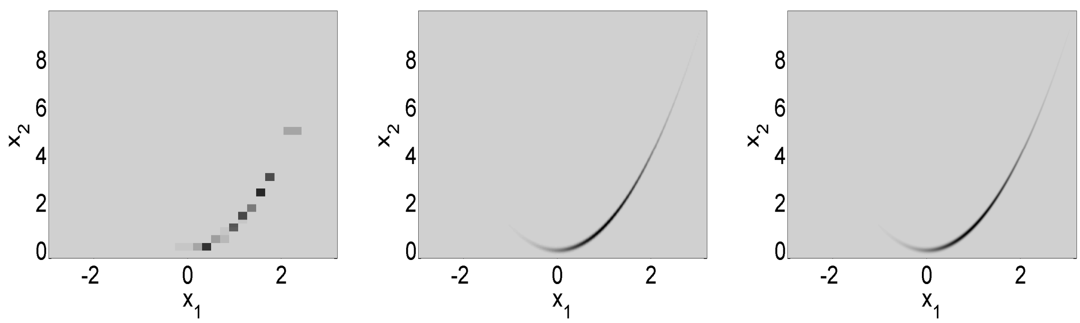
Figure 5.4: The 2D-Gaussian density, its contour plot and approximated functions with $\Lambda = 100, 10000, 1000000$.



(a) The 2D-Rosenbrock shape.



(b) Exact density.



(c) $\Lambda = 100$.

(d) $\Lambda = 10000$.

(e) $\Lambda = 1000000$.

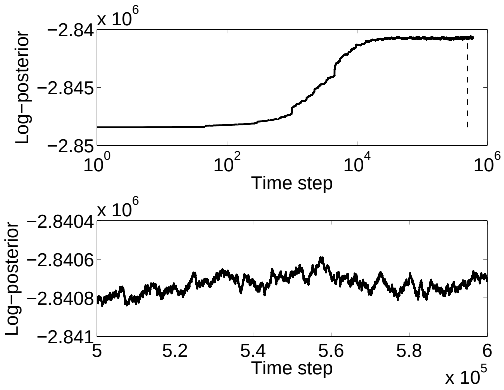
Figure 5.5: The 2D-Rosenbrock shape, contour plots of the density and its approximated functions with $\Lambda = 100$, 10000, 1000000.

5.2.3 Two Examples

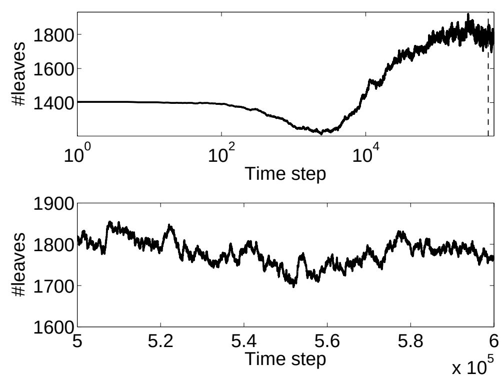
Our simple GR diagnostics was successful for the uniform densities as shown in 5.1, but was misleading for our approximated multivariate Gaussian and Rosenbrock densities. Here, we used the strategy discussed in Section 4.3.5 to obtain a reasonable starting state to initialise our Metropolis-Hastings chain $S(t)$. We then monitored the log-posterior and the number of leaves of $S(t)$ to assess convergence heuristically and get reasonable posterior mean histogram estimates.

Figure 5.6 shows the posterior mean histogram and trace plots obtained from a MCMC run based on $n = 1000000$ for the 2D-Gaussian approximated by $\Lambda = 10000$ leaves. We start the chain with an initial condition obtained from Algorithm 7 of Chapter 4 and monitor the log-posteriors and number of leaf nodes thereafter to obtain a suitable burnin time step. We observe the IAEs of the current states and the current averaged states in Figure 5.6(c) and note an improvement in the IAEs as averages are taken over states collected after burnin. This can be attributed to a smoothing effect. The addition of states from the desired stationary distribution allows us to go deeper into the space to obtain states that can better approximate the target function, but that might never be visited by a typical `RPQ` run. 5.6(e) shows the best histogram estimate (in terms of IAE) obtained from `RPQ` over $\tilde{S}_n$ that has an IAE of 0.0625 and 3022 leaf nodes. This histogram from `RPQ` has more splits occurring at regions with higher densities, which resulted in a sharper peak compared to the posterior mean histogram of Figure 5.6(d) that has 6022 leaf boxes and an IAE of 0.0502. We also observe that regions with lower densities are better described in the posterior mean histogram estimate whilst the histogram from `RPQ` is flatter at lower density regions, unable to capture details that the posterior mean histogram estimate managed to.

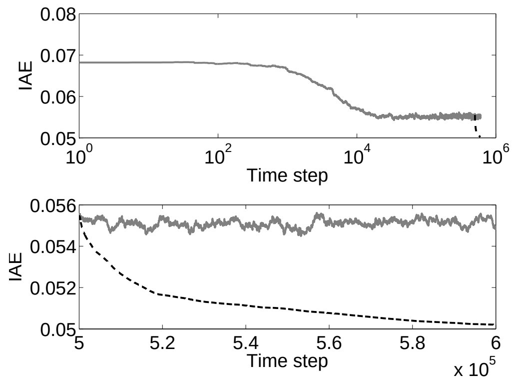
We now look at another MCMC run based on $n = 1000000$ for the 2D-Rosenbrock density approximated by $\Lambda = 10000$ and show the corresponding trace plots and posterior mean histogram estimate in Figure 5.7. Once again we observe an improvement in the IAEs as averages are taken over samples collected after the burnin (Figure 5.7(c)). The resulting posterior mean histogram estimate is shown in Figure 5.7(d) and has 14379 leaf boxes with an IAE of 0.0789; while the histogram from `RPQ` is shown in Figure 5.7(e) and has 13884 leaf boxes and an IAE of 0.1196. We observe that the averaged histogram managed to capture more information at the tail around the region $[1, 2] \times [8, 9]$ compared to the histogram from `RPQ`. Moreover, there are fewer sharp spikes in the posterior mean histogram estimate compared to the estimate from `RPQ`, implying a smoothing effect is taking place as samples are collected for averaging.



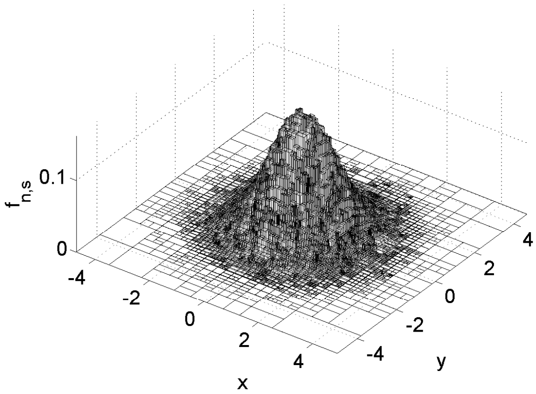
(a) Trace plots for number of leaf nodes.



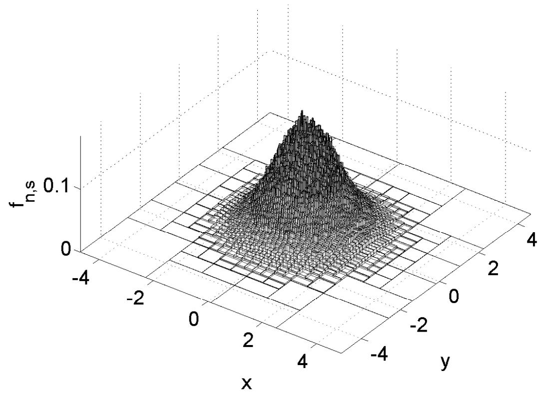
(b) Trace plots for log-posterior.



(c) IAEs for current (solid) and averaged states (dashed).

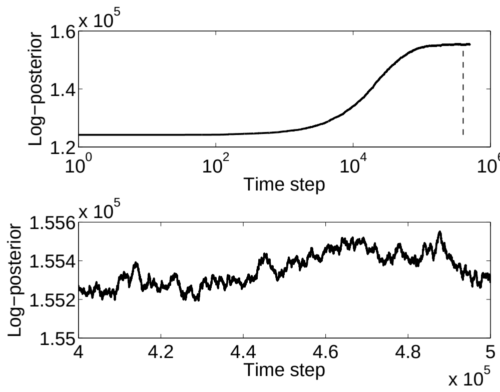


(d) Posterior mean histogram estimate.

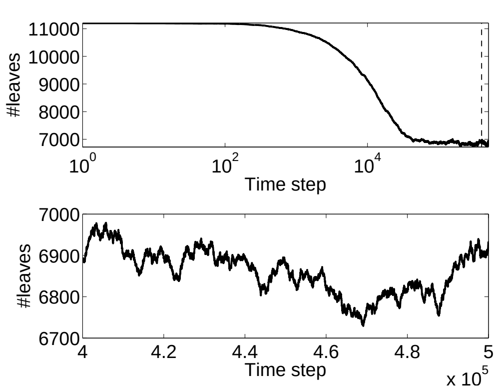


(e) Histogram estimate from `RPQ`.

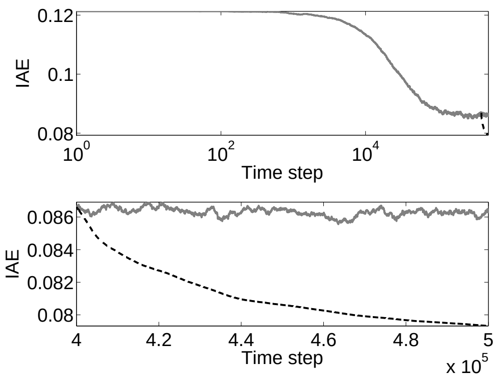
Figure 5.6: A MCMC chain for an approximated 2D-Gaussian density initialised by a state with 1404 leaf nodes. 10000 samples are collected with a thinout of 10 after burnin at time step 500000. The resulting posterior mean histogram estimate has 6022 boxes with IAE = 0.0502; whilst the optimal histogram estimated has 3022 boxes with IAE = 0.0625.



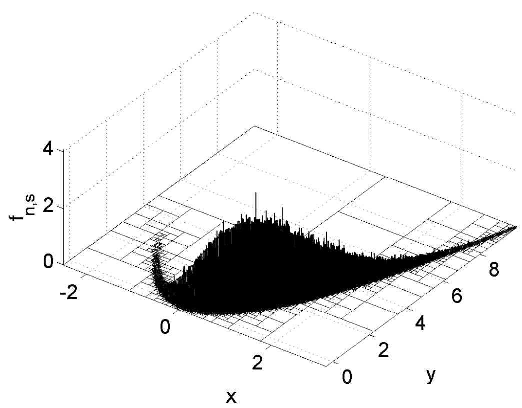
(a) Trace plots for log-posterior.



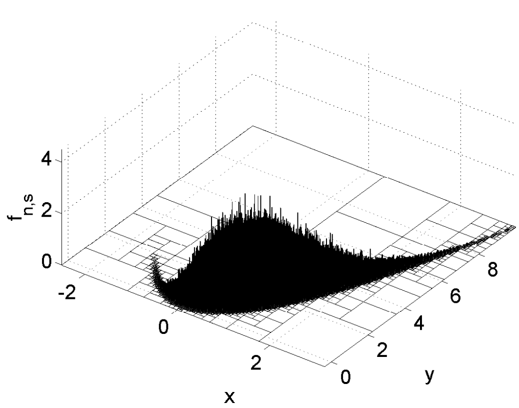
(b) Trace plots for number of leaf nodes.



(c) IAEs for current and averaged states.



(d) Posterior mean histogram estimate.



(e) Histogram estimate from `RPQ`.

Figure 5.7: A MCMC chain started for an approximated 2D-Rosenbrock density by a state with 11200 leaf nodes. 10000 samples are collected with a thinout of 10 after burnin at time step 400000. The resulting posterior mean histogram estimate has 14379 boxes with IAE = 0.0789; whilst the optimal histogram estimated has 13884 boxes with IAE = 0.1196.

5.2.4 Simulation Results

We now proceed with simulation studies for approximated functions of the Gaussian density in dimensions 1D, 2D, and 5D, and approximated functions of the Rosenbrock density in dimensions 2D and 5D. We use $\Lambda = 100, 10000, 1000000$ to approximate each density. For each density and its approximation, and for each n, we record both the IAE of the posterior mean histogram estimate (the first row) and the IAE of the best histogram estimate from `RPQ` over $\tilde{S}_n$ (the second row).

From Table 5.2 it is clear that as the sample size n increases from 10^2 to 10^7 in multiples of 10 the MIAE decreases in each Λ-specific approximation of the multivariate Gaussian and Rosenbrock target densities, for both the estimates from the MCMC run and `RPQ`. Once again, we simulate n data points from each Λ-specific approximation given by an SRP histograms and efficiently compute the IAE of our histogram estimates. We get the MIAE and standard error from 25 replications.

The MIAEs of the posterior mean histogram estimates are generally lower than the MIAEs of the optimal histogram estimates from the `RPQ`. For the 5D-Gaussian approximated by $\Lambda = 1000000$, the MIAE value of the posterior mean histogram estimate is higher than the MIAE value of the `RPQ` histogram when $n = 10^4$. However as n increases, the MIAE values of the posterior mean histogram estimate decrease steadily and eventually perform better than the `RPQ` histogram. We also observe that the MIAE values for the estimates of the 5D-Rosenbrock approximated by $\Lambda = 100$ across all n's are significantly smaller compared to the other MIAE values for the various approximated densities. Recall that the sample points used to produce the estimates are simulated from the approximated target distributions, and not the actual density. Thus, we are measuring the L_1-distance between the estimates and the approximated target densities. The approximation of the 5D-Rosenbrock density using $\Lambda = 100$ only has 1 leaf node where most of the mass are contained, whilst the remaining 99 boxes have mass near zero or zero. Recall that the Λ-specific approximations are weighted uniform mixtures over SRP partitions. Consequently, the 5D-Rosenbrock approximation using $\Lambda = 100$ is almost similar to a standard uniform density with support being the box that contains the bulk of the mass. Thus, as observed in the simulation studies for uniform densities in Table 5.1), the estimates obtained from samples simulated from the 5D-Rosenbrock approximation with $\Lambda = 100$ have lower MIAE values when compared to the MIAE values of the other approximations.

This phenomenon highlights the need for more leaf boxes to better approximate the target densities as d increases. Besides, as observed in Figure 5.2, the Hellinger distance of the λ-specific approximation for the Rosenbrock density is much larger than that for the Gaussian density for each Λ. Thus, the Rosenbrock density has a higher structural complexity than the Gaussian density in terms of requiring an SRP histogram with more leaf boxes or Λ to approximate it to a given Hellinger distance. We ensure that when $\Lambda = 1000000$ the Hellinger distance to Gaussian or Rosenbrock density is well below 0.05 in all our simulation experiments. The MIAE values clearly show that the dimension as well as the structural complexity of the density determines the performance of our estimator.

		Standard Gaussian densities			Rosenbrock densities	
Λ	n	1D	2D	5D	2D	5D
10^2	10^4	0.0568 (0.0057)	0.0826 (0.0076)	0.0555 (0.0076)	0.0322 (0.0053)	0.0054 (0.0077)
		0.0785 (0.0057)	0.1340 (0.0073)	0.1221 (0.0208)	0.0875 (0.0157)	0.0011 (0.0024)
	10^5	0.0256 (0.0019)	0.0231 (0.0022)	0.0152 (0.0023)	0.0102 (0.0022)	0.0009 (0.0012)
		0.0381 (0.0028)	0.0565 (0.0030)	0.0471 (0.0115)	0.0395 (0.0074)	0.0010 (0.0024)
	10^6	0.0083 (0.0003)	0.0075 (0.0006)	0.0038 (0.0004)	0.0032 (0.0009)	0.0002 (0.0003)
		0.0176 (0.0011)	0.0235 (0.0019)	0.0175 (0.0053)	0.0166 (0.0028)	0.0010 (0.0025)
	10^7	0.0025 (0.0002)	0.0023 (0.0002)	0.0011 (0.0001	0.0023 (0.0002)	0.0001 (0.0001)
		0.0073 (0.0008)	0.0097 (0.0012)	0.0060 (0.0021)	0.0088 (0.0021)	0.0007 (0.0013)
10^4	10^4	0.0583 (0.0054)	0.1650 (0.0042)	0.5137 (0.0070)	0.3620 (0.0111)	0.3186 (0.0106)
		0.0797 (0.0070)	0.1940 (0.0044)	0.5409 (0.0059)	0.4407 (0.0124)	0.6710 (0.0221)
	10^5	0.0274 (0.0015)	0.0826 (0.0076)	0.2509 (0.0026)	0.1821 (0.0053)	0.0885 (0.0030)
		0.0394 (0.0023)	0.1118 (0.0015)	0.2868 (0.0026)	0.2395 (0.0102)	0.2306 (0.0033)
	10^6	0.0128 (0.0004)	0.0508 (0.0005)	0.0771 (0.0009)	0.0797 (0.0006)	0.0262 (0.0006)
		0.0198 (0.0010)	0.0627 (0.0006)	0.1316 (0.0016)	0.1196 (0.0006)	0.1063 (0.0015)
	10^7	0.0060 (0.0002)	0.0256 (0.0003)	0.0220 (0.0004)	0.0267 (0.0003)	0.0080 (0.0003)
		0.0096 (0.0005)	0.0332 (0.0002)	0.0619 (0.0015)	0.0583 (0.0003)	0.0491 (0.0008)
10^6	10^4	0.0565 (0.0053)	0.1673 (0.0046)	0.6467 (0.0051)	0.3717 (0.0103)	1.0190 (0.0059)
		0.0790 (0.0070)	0.1940 (0.0058)	0.5409 (0.0059)	0.4483 (0.0134)	1.0363 (0.0043)
	10^5	0.0274 (0.0011)	0.0932 (0.0002)	0.4656 (0.0020)	0.1982 (0.0067)	0.7250 (0.0011)
		0.0397 (0.0022)	0.1129 (0.0017)	0.4730 (0.0014)	0.3518 (0.1044)	0.7574 (0.0010)
	10^6	0.0129 (0.0006)	0.0533 (0.0005)	0.3257 (0.0006)	0.1102 (0.0006)	0.4745 (0.0088)
		0.0199 (0.0011)	0.0646 (0.0005)	0.3278 (0.0008)	0.1423 (0.0006)	0.4953 (0.0009)
	10^7	0.0060 (0.0001)	0.0304 (0.0002)	0.2292 (0.0034)	0.0795 (0.0006)	0.3303 (0.0003)
		0.0097 (0.0005)	0.0413 (0.0003)	0.2483 (0.0006)	0.0823 (0.0006)	0.3532 (0.0004)

Table 5.2: MIAE (std. err.) for n samples from approximated 1D-, 2D- and 5D-Gaussian densities, 2D- and 5D- Rosenbrock densities. 10000 samples collected.

Chapter 6

Arithmetic for Spatio-Temporal Trajectories

This is joint work with Dr. Kenneth Kuhn, formerly from the Civil and Natural Resources Engineering Deparment, University of Canterbury, and Dr. Raazesh Sainudiin. The journal article version of this chapter has been published in the Journal of Aerospace Computing, Information and Communication (Teng et al., 2012).

6.1 Introduction

Air traffic controllers monitor the positions of aircraft and offer pilots guidance to ensure safe and efficient operations. Aviation systems researchers are developing decision-support tools to assist controllers as they manage increasing numbers of aircraft. Much research is framed around analysing and forecasting the locations of aircraft in space and time. Aircraft trajectory data are often investigated in the context of other data: for instance airspace configuration (air traffic control) data or weather data. Data regarding aircraft positions can be used to estimate, and help control, air traffic controller workloads, local environmental impacts (e.g., noise), airport or airspace throughput, the proximity of different aircraft trajectories, etc..

A wealth of aircraft position data is currently being produced. For instance, in the United States the Federal Aviation Administration (FAA) records precise data on the latitude, longitude, altitude, and assorted other data for aircraft in radar range at least every 12 seconds in en-route areas and every 4.2 seconds around airport terminals (Lester and Hansman, 2007). Data are expected to be produced at a higher frequency as technologies like automatic dependent surveillance-broadcast come into widespread use. At the same time, there are and will be increasing numbers of aircraft and other airborne objects to track.

Kuhn (2008) previously studied aircraft position data recorded at one FAA control centre over the course of 40 days. The data set takes up roughly 14 GB of space. In this same study, the author was interested in analysing the impacts of weather on aviation, and thus collected weather data for the 40 days of interest that take up another 10 GB of space when stored in the efficient hierarchical data format (hdf5). The sizes of datasets containing information collected over extended periods of time, tracking large numbers of aircraft, are problematic. Decision makers and researchers interested in the monitoring of real-time operations in particular face a challenge: how to quickly analyze and automatically summarize more data than can be stored in the primary memory of computers.

There is a need for a method to compactly store aircraft position and related data in a format that enables speedy analyses. Some aggregation of available data will be required, but data loss due to aggregation should be minimized. There should also be techniques for performing computationally efficient mathematical operations on aggregations of different datasets. We propose using SRPs to represent aircraft trajectory data, and perform arithmetic operations with them. For any given flight-specific trajectory data, we can use a SRP to partition a section of airspace (which minimally bounds the trajectory data) such that the finest cell in the partition is about the size of the aircraft. Only cells at the finest resolution are allowed to contain exactly one data point. The remaining cells with sizes larger than or equal to the finest resolution are empty with no data points. Each cell that contains a data point represents the position of a specific aircraft in airspace over a particular time period, whereby more computational resources are associated with areas where an aircraft is observed. Our proposed SRP data structure for flight-specific trajectory data can thus be thought of as a binary space-partitioning tree whose leaves are $\{0,1\}$-valued in order to represent the trajectory of a particular aircraft in a given region of the airspace over some time period. When the $\{0,1\}$-valued SRP for individual flight trajectories is a tree-based partition of the airspace within the radar's range, such that the leaf cells of the tree are either 0 or 1 to indicate the absence or presence of a particular flight, respectively, over a given time period, we can exploit the recursive properties of trees to perform computationally efficient arithmetic operations over the space of flight-specific trajectories. For instance, we can perform the addition of individual flight trajectories that are in the airspace over the same period of time in order to obtain the aggregate cotrajectories in the airspace during this period of time. Such an addition operation amounts to data aggregation of individual SRP trajectories into an aggregate SRP cotrajectory for which the leaves are allowed to have possibly more than one flight in them, depending on the length of the time period. Other arithmetic operations such as subtraction of two SRPs and multiplication of an SRP by a real number can also be performed. Thus, we can condense massive data into memory-efficient SRPs and perform subsequent arithmetic operations over them for aiding downstream decisions such as cotrajectory classification with additional weather data, fine-scale pollution monitoring, etc. We do not engage in such down-stream decisions using SRPs in this paper and focus instead on the foundations of SRPs for trajectory and cotrajectory arithmetics.

We note that there are many ways in which the flight observations from radar can sequentially enter a data structure for analysis. Suppose n $\mathbb{R}^d$-valued observations $X_1, X_2, \ldots, X_n$ sequentially enter the structure. We consider the following sequential entry settings in this study.

- In the n-presensed setting, all available data can sequentially enter the structure as one burst of all n points. In other words, all of the data are available in external-memory (or radar buffer).

- In the $n_{1:m}$-presensed setting, all available data can sequentially enter the structure as m bursts of $n_1, n_2, \ldots, n_m$ points from external-memory at conveniently choosable CPU times $t_1 < t_2 < \ldots < t_m$.

- In the $n_{1:m:\ldots}$-sensing setting, the CPU times $t_1 < t_2 < \ldots < t_m$ are driven by the real time line with yet unsensed burst indices beyond current CPU time t_m. We want our methods to computationally cope with data bursts and aid in decision-making in this more realistic online or dynamic sensor setting. We can obtain an idea of the limitations of our methods in the $n_{1:m:\ldots}$-sensing setting by first studying the memory and CPU limitations in the n-presensed and $n_{1:m}$-presensed settings.

Our proposed data structure is capable of handling both the $n_{1:m}$-presensed and $n_{1:m:\ldots}$-sensing settings. Thus, our dynamic data structure grows with each new burst from the radar and shrinks with each landing event in order to efficiently represent all aircraft positions in airspace at the present time or some specified interval of time. This is elaborated further in Section 6.5.

6.2 Related Work

A number of research efforts have been put into investigating large aircraft trajectory data sets. An interactive visualization tool called FromDaDy (Hurter et al., 2009) was developed for exploratory data analysis of aircraft trajectories and efficient detection of specific features. In this sophisticated work, an aircraft trajectory is a single line, or more precisely, dots connected by a line due to discrete observations by radar detection. There are hence no duplications of trajectories for a flight unit, but rather the trajectories are spread across views (Hurter et al., 2009). Using lines as the unit object enables the user to "filter, remove and add trajectories in an iterated manner until they extract a set of relevant data" (Hurter et al., 2009) by using brush, pick, and drop techniques for selection. Boolean operations such as union (or addition) or intersection of the lines can be performed efficiently as well. This aspect is comparable to our data structure where each SRP object represents an aircraft trajectory over a time interval such that efficient aggregation (union) or intersection operations may be performed over SRPs. Recall that our SRP objects can be thought as a $\{0, 1\}$-valued structure such that an aggregation of these objects in integer arithmetic would also return the total number of aircraft in any given cell over the specified period of time. Using SRPs we are not only able to aggregate the individual trajectories, but also perform 1) query operations over a cuboidal region of the airspace over some time period, and 2) arithmetic operations over the aggregated trajectories for aiding downstream decisions. However, FromDaDy is a powerful data visualization tool and more efficient than SRPs in terms of trajectory selection, whereby the selection shape is not restricted by rectangular query boxes. This is due to the brushing technique that allows for geometrical queries that are more complex in shape than the cuboids of SRPs. We think that FromDaDy of Hurter et al. (2009) and our SRP objects nicely complement each other in terms of combining visualization techniques with arithmetic techniques for trajectories.

Wilkerson et al. (2010) presents results on the geographic distribution of aircraft carbon dioxide emissions by using radar track data of the type analyzed here, as well as other data regarding the trajectories of aircraft flying through areas where radar data were unavailable. The authors were able to access data on trajectories over the course of 24 hour periods but note that "the large daily files are too large and cumbersome to load into computer memory" (Wilkerson et al., 2010). The authors aggregated the data using a grid-based approach that is considered later in this paper. Authors investigating the aviation system impacts of adverse weather often use a similar grid-based approach. To analyze air traffic control system performance, the FAA in the United States uses the Weather Impacted Traffic Index (WITI) (Callaham et al., 2001). WITI uses a regular grid overlaid on the United States and then compares weather and aircraft trajectory data in each grid cell. Here, Callaham et al. (2001) notes the "trade-offs regarding fineness of gridding". The finer the gridding, the heavier the needs for space and computational time, while also facing "excess information representation" (Callaham et al., 2001). Conversely, if the grid is too coarse, one then runs into the issue of a lack of information representation. We thus suggest using SRPs, which allows the finest resolution to be at the level of the size of the aircraft while not having unneeded information. We will further explore this in Section 6.6.

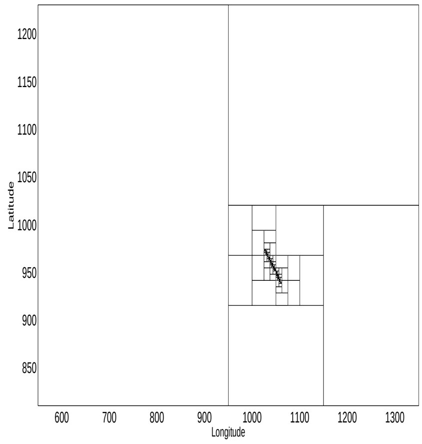
Other tree-based data structures are used by Agarwal et al. (2002) and Tao and Papadias (2005) to analyze moving objects, with the common aim of answering range queries efficiently, which typically involves getting the number of points (planes) that are contained in some query box (location). Variations and extensions of well-known R trees (R for rectangle) for spatial access and queries are used by Agarwal et al. (2002). Here, nearby points are grouped with their minimal bounding rectangle and each bounding box is known as a region. The bounding boxes or regions play a main role in deciding if there is a need to descend into the subtrees. The authors augment the R-trees with summarized information, for instance the total number of points, for each box of the tree. The authors then introduced a temporal aspect such that the query is now "the number of points in a given box for a given time interval." If we are given T timestamps, then for a given region, the corresponding summarized information associated with each time stamp is now stored in a B tree. An aggregate multi-version R-B tree is developed to perform queries for moving objects. The term "aggregate" here is used to describe the augmentation of the summarized information at each box of the tree. A new box is created when an object moves to another position that also changes the position and size of parent box. The summarized information are updated accordingly as well. The aggregate R-B tree, which holds summarized information at time stamp t is essentially a different object compared with our SRP tree that is a $\{0, 1\}$-valued structure for flight-specific trajectories. But, if we aggregate the flight-specific SRP objects, we obtain an SRP object that has summarized information of the trajectories, which is similar to the aggregate R-B tree. Now building the underlying R-tree, which is a balanced tree whereby all its leaf nodes have to be at the same height, can be a challenging task in terms of building an efficient tree such that a query search requires as few descents into the subtrees as possible. There is therefore a trade-off between information fineness and computational/memory needs for the data structure of Agarwal et al. (2002). Our SRP objects have the advantage of allowing us to summarize information at the finest resolution in airspace (i.e., each plane is enclosed by a box just big enough to fit it) such that we do not lose this crucial information during aggregation. Note that the aggregated SRP object at an instant of time t has to be a $\{0, 1\}$-valued structure to remain collision free, for if the value at a leaf box or cell is more than one at an instant of time, then it means that a collision has happened there. One of the main objectives of Tao and Papadias (2005) is the development of algorithms that maintain the median of a set of moving points for both the on-line (i.e., our $n_{1:m:\ldots}$-sensing setting) and off-line setting (i.e., our n-presensed or $n_{1:m}$-presensed settings). The algorithms of Tao and Papadias (2005) use kinetic kd trees on sets of moving objects for query tasks. Typically, a kd-tree is built by splitting the data at the coordinate-specific median into two subsets where splits are done by alternating at the coordinates. Two variants of kd-trees are proposed: a δ-pseudo kd-tree (which turns out to be an almost balanced tree) that allows for efficient insertion and deletion, and a δ-overlapping kd-tree (a perfectly balanced tree) where the bounding boxes of two children are allowed to overlap. Similar to the R-trees of Agarwal et al. (2002), the kd-trees of Tao and Papadias (2005) have nodes that already contain summarized information and are essentially different from our basic SRP objects that are specific to a flight instance.

The main motivation of the work of Agarwal et al. (2002) and Tao and Papadias (2005) is to seek efficient data structure for query tasks. We are also able to perform query tasks with our data structure by intersecting a query box with the aggregated SRP object and extract the needed information. However, this querying aspect of SRPs is not explored in this study since the main focus of our study is to develop memory-efficient tree-based data structures for individual and aggregate flight trajectories for purposes of arithmetical operations over them. Finally, we would like to emphasize that our SRP trees are uniquely suited to perform efficient arithmetic operations, especially on a set of aggregated SRP objects for, say, cotrajectory pattern analyses, especially in conjunction with local weather data. We think that the data structures of Hurter et al. (2009), Agarwal et al. (2002) and Tao and Papadias (2005) can be used in complementary ways with SRPs, for which the real strength lies in performing arithmetic operations over the space of trajectories and cotrajectories, to comprehensively deal with downstream decision problems using our SRP arithmetics bolstered by interactive visualization of Hurter et al. (2009) as well as fast querying and coarse aggregation in Agarwal et al. (2002) and Tao and Papadias (2005).

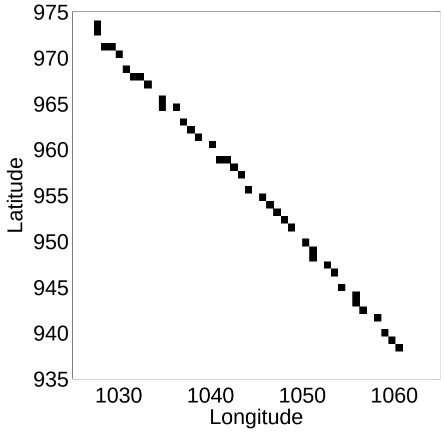
6.3 SRPs and Flight Trajectories

We will now apply SRPs to the analysis of aircraft trajectories. Let $X_1, \ldots, X_n$ be the time-ordered aircraft position data provided by FAA radar facilities. Data are typically provided as ordered high-dimensional tuples containing position data in $\mathbb{R}^2$ or $\mathbb{R}^3$, e.g., (latitude, longitude) or (latitude, longitude, altitude), and related data regarding aircraft heading, speed, type, etc. We will focus on position data to simplify discussion and because position data are the most relevant for many applications. Here, we are interested in constructing an SRP by recursive bisections to represent position data. A bisection only happens when a node has more than one point and this bisected node will not produce child nodes that have volume less than a predefined minimum volume $\lambda^* = 2^{-i^*} \operatorname{vol}(\boldsymbol{x}_{\rho v})$, where $i^* = \lfloor \log_2 \lambda^{-1} \operatorname{vol}(\boldsymbol{x}_{\rho v}) \rfloor$, and λ is taken to be an approximate volume of the aircraft. This splitting criteria guarantees that a leaf box with volume more than λ^* will not have any points in it and leaf boxes with volume λ^* may have more than one observation in them. The resulting SRP that encloses time-ordered aircraft position data of a particular aircraft is then an SRP trajectory. Here we will use the terms SRP and SRP trajectory interchangeably. The procedure to get an SRP trajectory is shown in Algorithm 8.

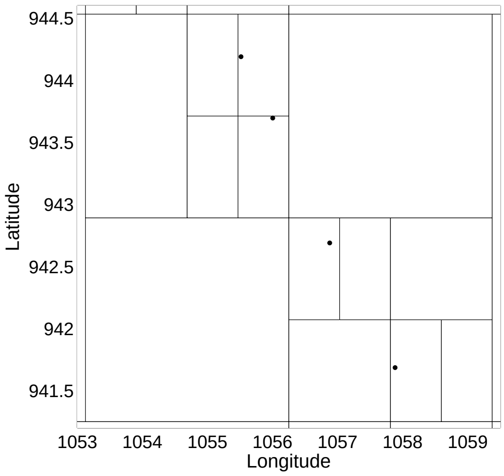
Figure 6.1(a) shows the SRP trajectory with root box $[810, 1230] \times [550, 1350]$ for aircraft position data $X_1, \ldots, X_n$, while Figure 6.1(b) shows the shaded boxes in which points fall into. We zoom into Figure 6.1(a) to obtain Figure 6.1(c) and its corresponding tree in Figure 6.1(d). The aircraft at some position X_j is shown as a black point in Figure 6.1(c) and is enclosed by a box with volume λ^*. We note that the observations are discrete in time, hence resulting in boxes with points that are disconnected.



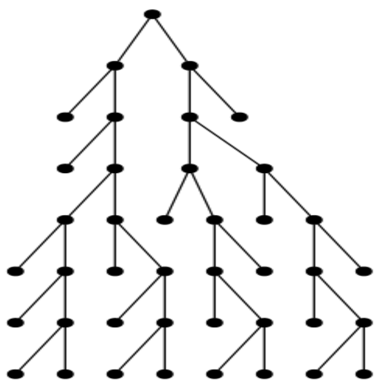
(a) SRP trajectory for aircraft position data.



(b) Shaded boxes in the SRP trajectory.



(c) Aircraft positions enclosed by boxes.



(d) The tree corresponding to (b).

Figure 6.1: An SRP trajectory for aircraft position data and its corresponding tree.

Algorithm 8: $s =$ `makeSRP`$(\{X_1, \ldots, X_n\},\ \lambda^*,\ \boldsymbol{x}_\rho)$

input :

1. data $\{X_1, \ldots, X_n\} \subseteq \mathbb{R}^d$;
2. minimum volume λ^*;
3. root box $\boldsymbol{x}_\rho$.

output : an SRP s.

Make a new node s with box $\boldsymbol{x}_\rho$ and $\#\boldsymbol{x}_\rho \leftarrow 0$

for $j = 1 : n$ **do**

  `insertData`$(\rho,\ X_j,\ \lambda^*)$ `// (see Algorithm 9)`

end

return s

Algorithm 9: `insertData`$(\rho v, X_j, \lambda^*)$

input :

1. node ρv,
2. data $X_j \in \boldsymbol{x}_{\rho v}$,
3. minimum volume λ^*.

if *box* $\boldsymbol{x}_{\rho v}$ *contains* X_j **then**

  Increment $\#\boldsymbol{x}_{\rho v}$ by 1

  if *(ρv is a leaf node)* & $(\frac{1}{2} \cdot \mathrm{vol}(\boldsymbol{x}_{\rho v}) \geq \lambda^*)$ **then**

    Make left node $\rho v\mathsf{L}$ with box $\boldsymbol{x}_{\rho v \mathsf{L}}$

    Make right node $\rho v\mathsf{R}$ with box $\boldsymbol{x}_{\rho v \mathsf{R}}$

    $\#\boldsymbol{x}_{\rho v \mathsf{L}} \leftarrow 0, \#\boldsymbol{x}_{\rho v \mathsf{R}} \leftarrow 0$

    Graft onto ρv as left child the node $\rho v\mathsf{L}$ and `insertData`$(\rho v\mathsf{L},\ X_j,\ \lambda^*)$

    Graft onto ρv as right child the node $\rho v\mathsf{R}$ and `insertData`$(\rho v\mathsf{R},\ X_j,\ \lambda^*)$

  end

  if *ρv is not a leaf node* **then**

    `insertData`$(\rho v\mathsf{L},\ X_j,\ \lambda^*)$

    `insertData`$(\rho v\mathsf{R},\ X_j,\ \lambda^*)$

  end

end

6.4 Aggregations and Operations with SRP Trajectories

We can add/subtract SRP trajectories to produce an aggregate SRP trajectory by using `AddSRPHist` of Algorithm 6 in Chapter 4, though, instead of adding/subtracting heights of nodes, we will add/subtract the counts at each node, as illustrated in Figure 6.2. This would, for instance, allow us to come up with aggregate frequency histograms when different data sets or even points have different levels of precision. This is interesting from the point of view of aviation systems research, since aircraft size, aircraft equipage, and the fidelity of different relevant data streams are variable.

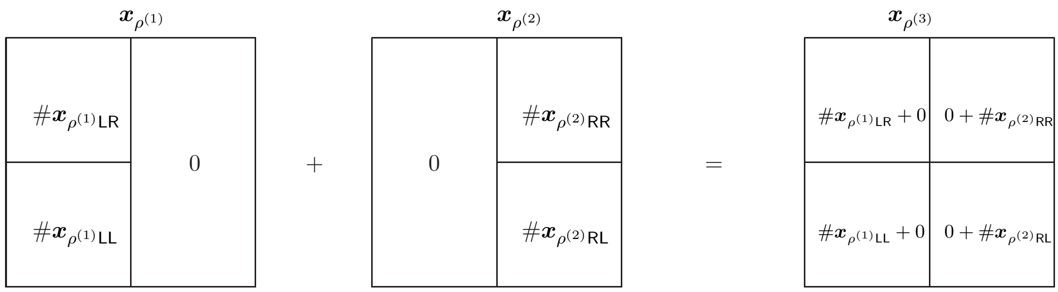


Figure 6.2: An addition operation between SRPs $s^{(1)}$ and $s^{(2)}$.

For a given SRP trajectory $s^{(1)}$ with root node $\rho^{(1)}$, define the scalar product $\alpha \cdot \rho^{(1)}$ to be the root node obtained from $\rho^{(1)}$ by transforming the count $\#\boldsymbol{x}_{\rho^{(1)}v}$ of every node $\rho^{(1)}v$ in $s^{(1)}$ to $\alpha \cdot \#\boldsymbol{x}_{\rho^{(1)}v}$. Now, for any real-valued α, β, we can obtain a linear combination $\alpha \cdot \rho^{(1)} + \beta \cdot \rho^{(2)}$ of two SRPs $s^{(1)}$ and $s^{(2)}$ with root nodes $\rho^{(1)}$ and $\rho^{(2)}$ by applying `AddSRP`$(\alpha \cdot \rho^{(1)}, \beta \cdot \rho^{(2)})$. When $\alpha = \beta = 1$, `AddSRP`$(\alpha \cdot \rho^{(1)}, \beta \cdot \rho^{(2)})$ is equivalent to an addition between the SRP trajectories $s^{(1)}$ and $s^{(2)}$. Figure 6.3 shows high fidelity aircraft position data points enclosed by SRPs for three flights to which we assign fictional flight numbers ABC123, DEF456, and GHI789. The top panel in Figure 6.3 shows the shaded leaf boxes, while the bottom panel shows the corresponding box boundaries at all nodes of the SRP trajectories. We give an example of adding three SRP trajectories in Figure 6.4. The aggregate SRP trajectory for these three flights is shown in Figure 6.4(a) as shaded leaf boxes and as box boundaries at all nodes in Figure 6.4(b).

It is easy to access information such as frequencies (box heights) and airspace locations (the positions of the boxes) since the SRP trajectories store such information. A simple visual inspection of an aggregate trajectory SRP can be useful: areas where boxes are smaller (or have darker shades) are where frequencies are larger and indicate sections of airspace that are more frequently occupied. We now process huge data sets containing the positions of

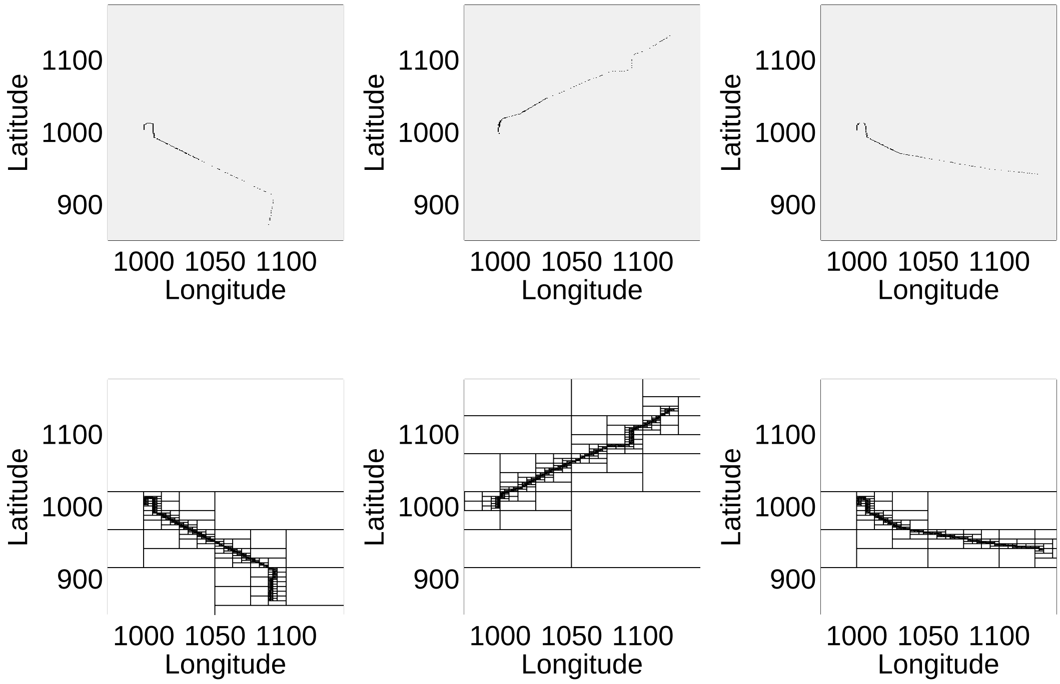
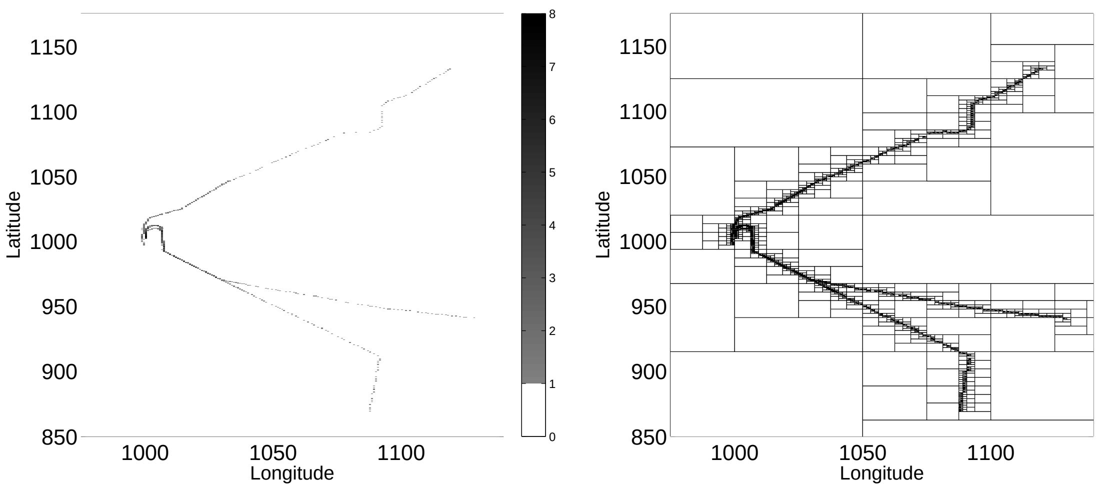


Figure 6.3: SRP trajectories of flights ABC123, DEF456, GHI789.



(a) Shaded boxes.

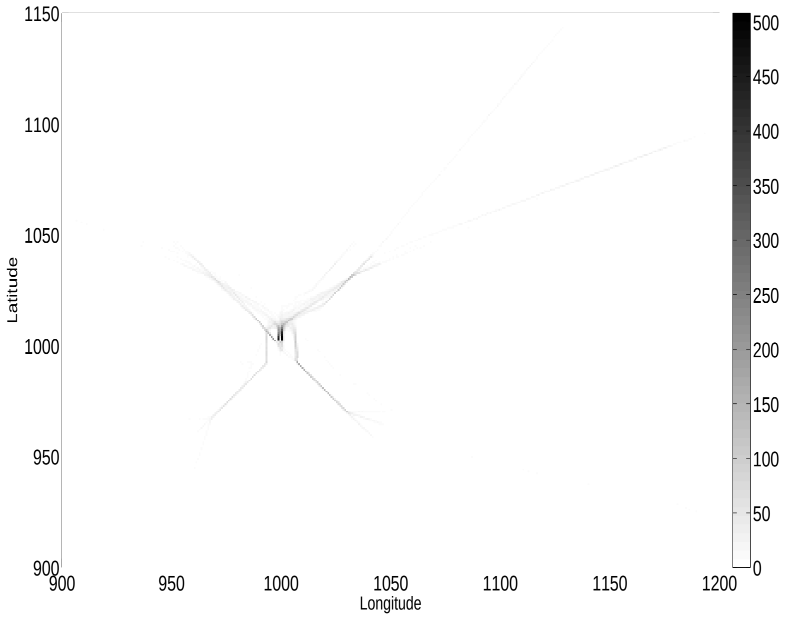
(b) Box boundaries at all nodes.

Figure 6.4: Aggregate SRP trajectory for the flight trajectories of Figure 6.3.

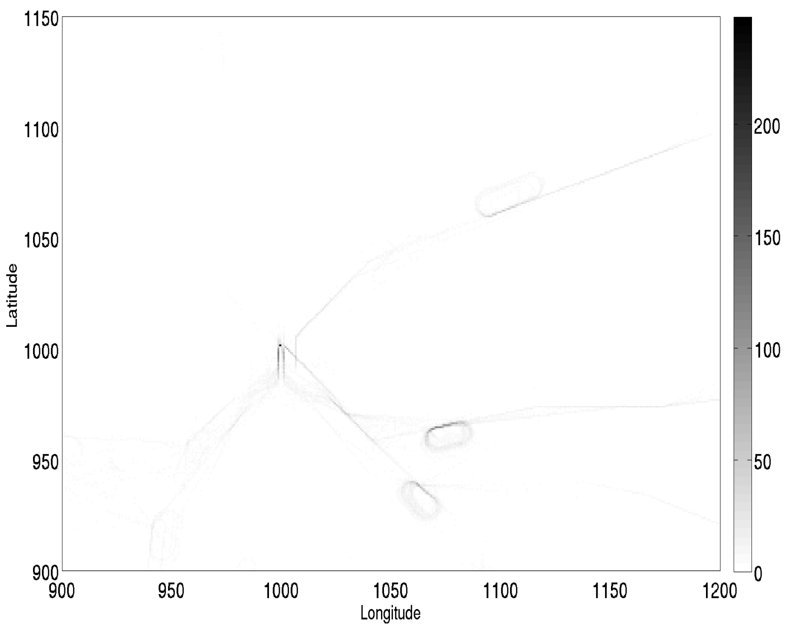
different aircraft at different levels of precision and quickly aggregate them selectively to obtain desired aggregate trajectories. For instance, such selective aggregations could be useful when analyzing air traffic patterns for different weather conditions. Figures 6.5(a) and 6.5(b) show aggregate SRP trajectories on arrivals at a busy North American airport for periods of 6 h during a day with good weather and another with bad weather, respectively. Latitude and longitude data have been converted to Cartesian coordinates with the airport located approximately at point (1000, 1000). Given that information can be accessed conveniently, it would be easy to, for example, manually/qualitatively or automatically/quantitatively compare Figures 6.5(a) and 6.5(b), and thus compare operations when weather conditions were benign vs when they were adverse.

We can further perform arithmetic on these aggregate SRP trajectories. By an abuse of notation, let $s^{(1)}$ denote the aggregate SRP trajectory of the good weather day and $s^{(2)}$ be the aggregate SRP trajectory of the bad weather day, each with root nodes $\rho^{(1)}$ and $\rho^{(2)}$, respectively. Now, let $\alpha = 1$ and $\beta = -1$. Then, $\texttt{AddSRP}(\alpha \cdot \rho^{(1)}, \beta \cdot \rho^{(2)})$ is equivalent to a subtraction between the two aggregate SRP trajectories, and the resulting aggregate SRP trajectory is able to provide information on which airspace will be more/less frequented on the good weather day compared with the bad weather day. This is illustrated in Figure 6.6, which shows the differences between the two weather aggregate SRP trajectories in Figure 6.5. Lighter areas indicate sections of airspace more frequently occupied on the bad weather day, whereas darker areas indicate locations of airspace frequented more often on the good weather day.

We have shown that SRPs can be used to represent and operate (linearly combine) sets of aircraft trajectories. Thus, further analysis of how SRPs like those shown in Figures 6.5(a) and 6.5(b) evolve over time and as weather patterns change is warranted. We note that bad weather days are generally different and it will likely prove quite difficult to classify and predict what will happen on a bad weather day. However, Figures 6.5 and 6.6 could be the first step toward classification and prediction of airspace usage based on concurrent weather data. There are potentially other applications of SRPs in other fields that require the analysis of object movements, e.g. animal migration tracking. Figure 6.7 shows animal track data for a single animal over three days represented by SRPs (data courtesy of Mr. Paul Sagar). Figure 6.7(a) shows the movement of the animal of each individual day represented by individual SRP trajectories, whilst Figure 6.7(b) is the aggregate SRP trajectory obtained by adding the three individual SRP trajectories. The core functions needed to implement the algorithms here can be found in a GNU general public licensed `C++` class library (Sainudiin and Harlow, 2010).



(a) A good weather day.



(b) A bad weather day.

Figure 6.5: Aggregate SRP trajectories on a good weather day and a bad weather day respectively.

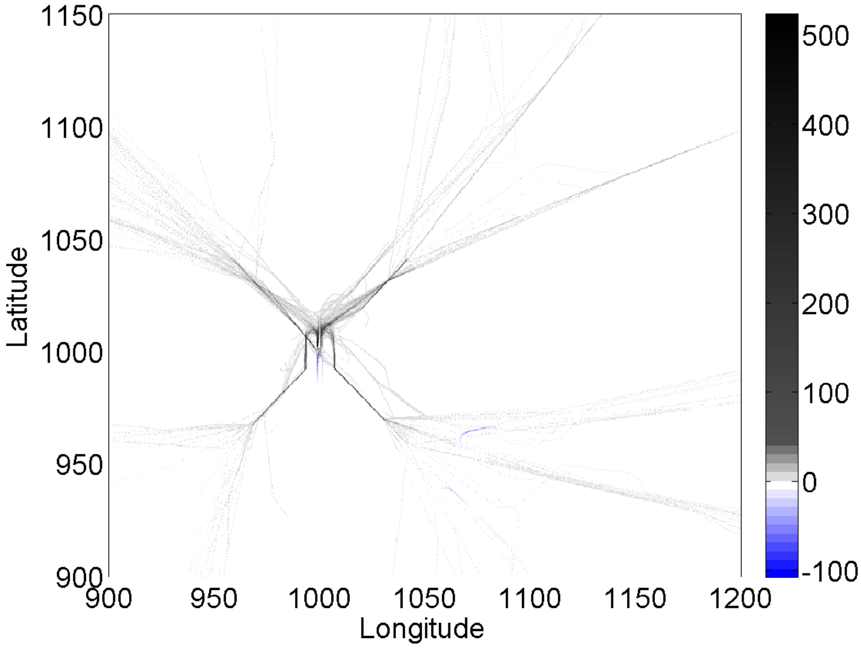
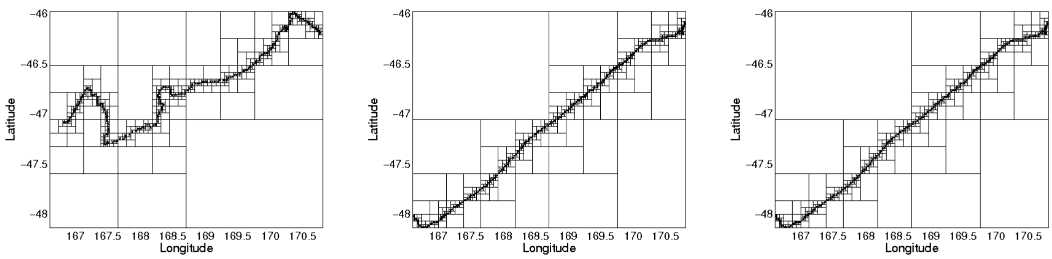
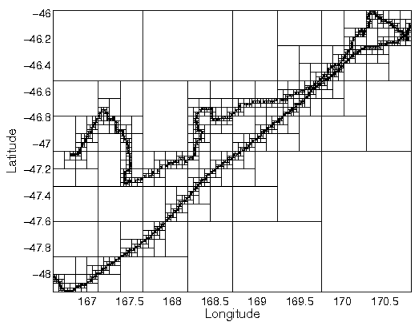


Figure 6.6: Subtraction between aggregate SRP trajectories on a good and bad weather day.



(a) Individual SRP trajectories.



(b) Aggregate SRP trajectory.

Figure 6.7: Using SRPs to track animal migration.

6.5 A Dynamic Airspace Data Structure with SRPs

6.5.1 Reuniting Nodes

If an (aggregate) SRP trajectory has pairs of sibling nodes that share the same properties, for instance, no points in each node, we can reunite these nodes to obtain its parent node and thereby reduce the size of the SRP to conserve memory. If two child nodes require two units of memory, then a reunification will halve the need for memory. Furthermore, the reunited node provides the same statistical information as its child nodes, so there is no loss of information. Figure 6.8 shows that boxes $\boldsymbol{x}_{\rho\mathsf{LL}}$ and $\boldsymbol{x}_{\rho\mathsf{LR}}$ can be reunited to get $\boldsymbol{x}_{\rho\mathsf{L}}$ since neither $\boldsymbol{x}_{\rho\mathsf{LL}}$ nor $\boldsymbol{x}_{\rho\mathsf{LR}}$ have any points in it. Algorithm 10 describes this procedure of reuniting empty nodes.

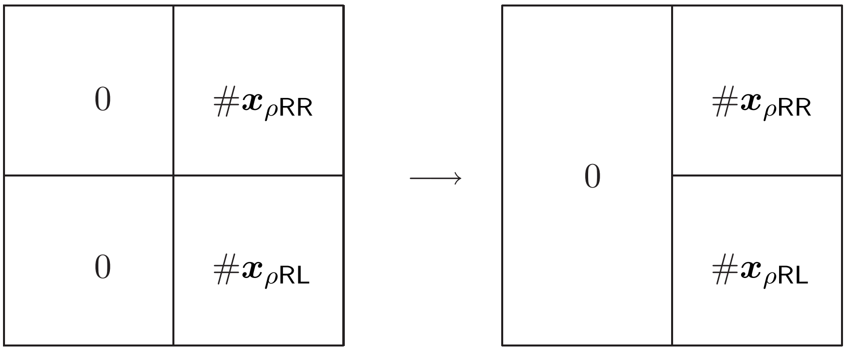


Figure 6.8: Reuniting boxes $\boldsymbol{x}_{\rho\mathsf{LL}}$ and $\boldsymbol{x}_{\rho\mathsf{LR}}$ to get box $\boldsymbol{x}_{\rho\mathsf{L}}$.

Algorithm 10: `ReuniteEmptyNodes`(ρv)

input : a node ρv

if *ρv has left child* **then**
 `ReuniteEmptyNodes`($\rho v\mathsf{L}$)
end
if *ρv has right child* **then**
 `ReuniteEmptyNodes`($\rho v\mathsf{R}$)
end
if *($\rho v\mathsf{L}$ and $\rho v\mathsf{R}$ are leaf nodes)* & *($\#\boldsymbol{x}_{\rho v\mathsf{L}} = \#\boldsymbol{x}_{\rho v\mathsf{R}} = 0$)* **then**
 $\#\boldsymbol{x}_{\rho v} \leftarrow 0$
 Delete the two child nodes $\rho v\mathsf{L}$ and $\rho v\mathsf{R}$
end

6.5.2 A Dynamic Airspace Data Structure

Say we are only interested in tracking flights that are currently in the airspace. In other words, in some time interval t, we only want information of aircraft that are still in flight and have no need for any information about aircraft that have already landed. We achieve this by constructing aggregate SRP trajectories that specifically describe the airspace for a sequence of time intervals. For example, Figure 6.9(a) shows the aggregate SRP trajectory of three aircraft in some time interval. One of the aircraft has already landed at the airport (represented by the black dot at point $[1000 \times 1000]$). At the next time interval, the trajectory of the landed flight is removed or subtracted from the aggregate SRP trajectory, producing empty leaf nodes that initially held information of the removed flight. Now using `ReuniteEmptyNodes`, we reunite any pair of sibling leaf nodes that are empty to obtain an aggregate SRP trajectory that only has information of the flights in the air, as seen in Figure 6.9(b). We thus obtain a dynamic airspace data structure where, at every time interval, the corresponding aggregate SRP trajectory only keeps information of the aircraft that are in the airspace. This is an instance of the $n_{1:m}$-presensed setting where trajectory data enters the structure at various timestamps. In other applications, another given property of the leaf nodes may be more appropriate than the property of being empty. In such cases, Algorithm 10 can be modified to reunite nodes with the given property.

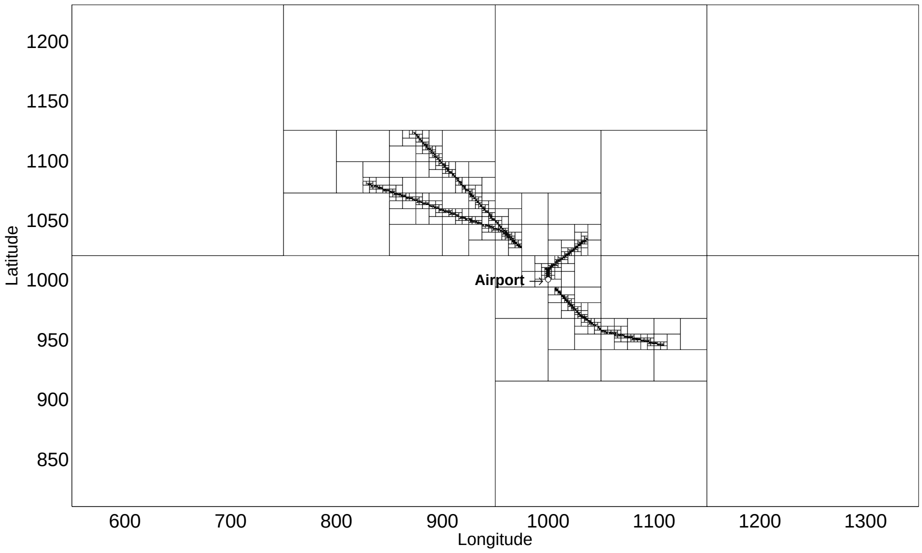
6.6 Complexity

We compare the space or memory as well as the time requirements of our data structures with those of a regular grid used by Callaham et al. (2001).

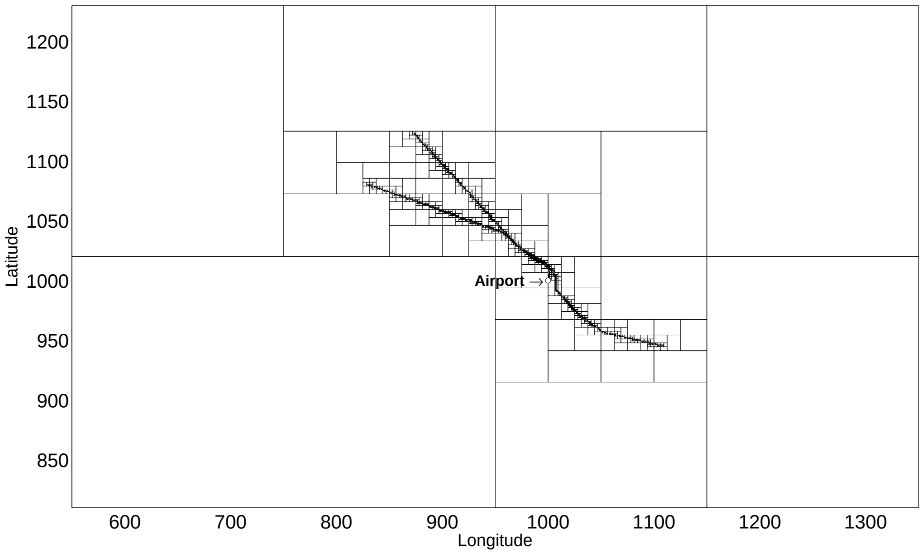
6.6.1 Space Complexity

The setup of the construction of SRP trajectories prevents unnecessary splitting on places without any flight visitations, i.e., as long as a box is empty, there is no need to bisect that box. This is in contrast to a regular grid construction where the root box is split until each cell has the same volume: in this case, λ, in order to represent aircraft position data. Note that we are storing recursively computable statistics for each box or cell. Thus, the number of boxes or cells reflects the memory requirements of the different data structures.

Suppose the regular grid is over a hypercube with box $\boldsymbol{x} = [0, 1]^d$. Let h be the side length of a cell in the grid along each of the d coordinates. Then, the volume of the cell is $h^d = \lambda$. Now let $m = 1/h$ be the number of cells along each coordinate. Then, there are m^d many

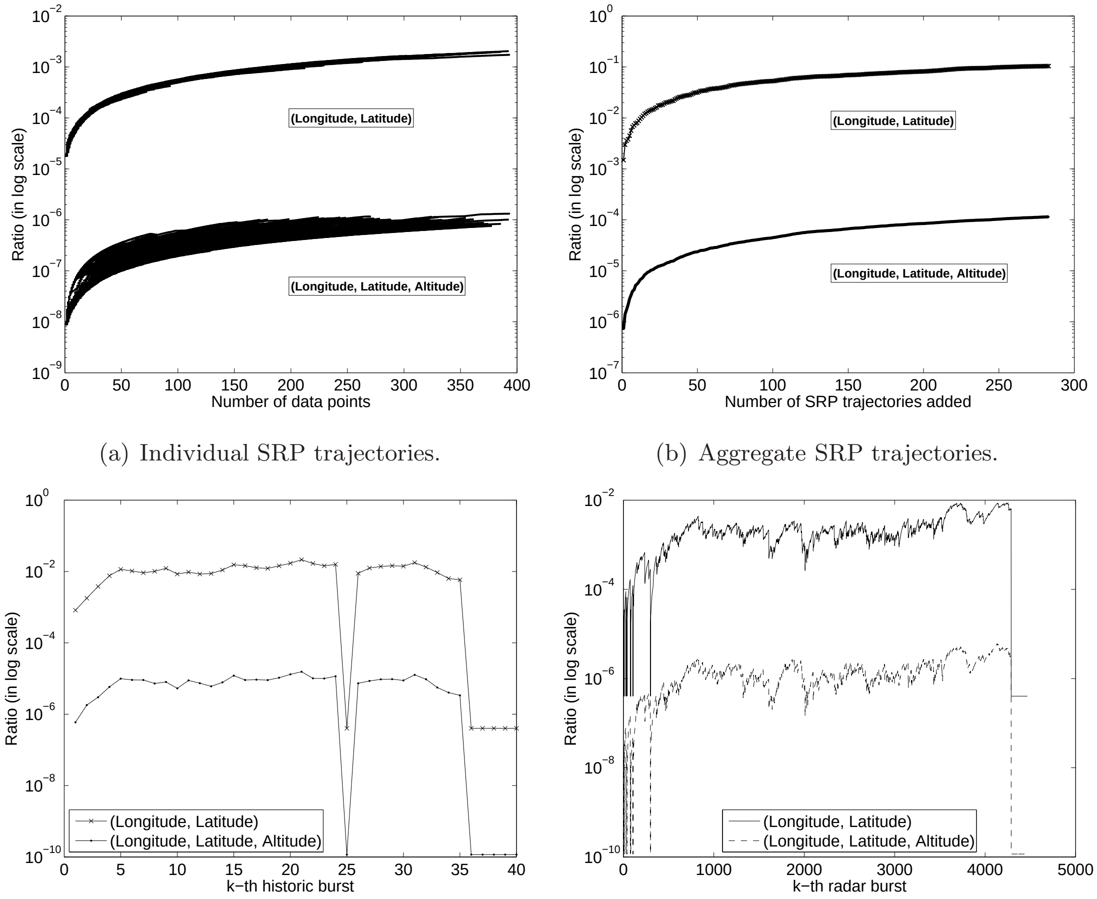


(a) A plane is just about to land.



(b) Trajectory of the landed plane removed.

Figure 6.9: A dynamic airspace data structure using SRPs.



(a) Individual SRP trajectories.

(b) Aggregate SRP trajectories.

(c) Dynamic airspace structure ($n_{1:m}$-presensed).

(d) Dynamic airspace structure ($n_{1:m:\ldots}$-sensing).

Figure 6.10: Ratio of the number of nodes in an SRP to the number of cells in a regular grid.

cells in a regular grid representing aircraft trajectory where

$$m^d = \left(\frac{1}{h}\right)^d = \frac{1}{\lambda} \ .$$

Thus, the memory requirements needed by a grid to represent aircraft position data grows exponentially with dimension d, resulting in the need for heavy computational storage requirements to produce data based on a grid. This is not desirable for massive data problems, especially in high dimensions.

The gain in memory for larger values of d is significant using SRPs as opposed to regular grids. In the graphs of Figure 6.10, the ratio of the number of nodes in an SRP data structure to the number of cells in a regular grid in shown in log scale. Figure 6.10(a) shows the ratios for SRP trajectories with coordinates (latitude, longitude) and (latitude, longitude, altitude), respectively. Figure 6.10(b) shows the ratios for aggregate SRP trajectories. Figure 6.10(c) gives the ratios for a dynamic structure for historic data blocked into 30 min and is in a $n_{1:m}$-

presensed setting, i.e., the k-th "historical" burst is from an interval of 30 min. Figure 6.10(d) has ratios for a dynamic structure constructed using data arriving from the radar or a $n_{1:m:\ldots}$-sensing setting. The kth time interval corresponds to the radar time for which we set at 10 s here. As d increases from 2 to 3, the ratio decreases by three orders of magnitude. Note that no planes were observed in the air space at the 25-th and 35-th historic bursts of Figure 6.10(c), causing the SRP trajectory to shrink back to its corresponding root box and consequently there is only one node count. This phenomenon is also observed from the "dips" seen in Figure 6.10(d).

Let the root box of the SRP trajectory be the same as that of the regular grid. The SRP trajectory requires $2j+1$ nodes to represent aircraft position data, where j is the total number of leaf nodes. As data are inserted into the SRP tree, the number of splits is determined by various factors, which include $\lambda = h^d$ (and thus the maximal depth $i^* = \lceil d \log_2(1/h) \rceil$ of the SRP), the number of data points, and interestingly, the position of the aircraft relative to its history in the SRP tree. For instance, fewer nodes are required to represent data that are clustered closely and/or if aircraft are repeatedly visiting the same areas. An example is given in Figure 6.11 for two SRP trajectories for position data of aircraft A and B. A flat plateau is observed between point 150 to point 280 for aircraft B in contrast to the fairly straight curve of aircraft A. A more detailed analysis shows that the flat plateau is a result of aircraft B circling around the vicinity of $[1100, 1075]$, i.e., points are likely falling into the same boxes due to repeated visits, and thus there are fewer splits of the data structure around that vicinity.

Now suppose n, the number of data points in our set of trajectories, is so large such that $n = (1/h)^d = m^d$, and that each cell in the grid contains at least one point. The corresponding SRP will then have $m^d - 1$ splits resulting in $2m^d - 1$ nodes, i.e., $\mathcal{O}(m^d)$, which is just as memory intensive as constructing a grid. This is, however, the worst-case scenario with a completely occupied airspace. For any data point that enters the SRP, at most, i^* number of splits are required for the point to be enclosed in a box of depth i^*. The number of nodes required in the SRP is then

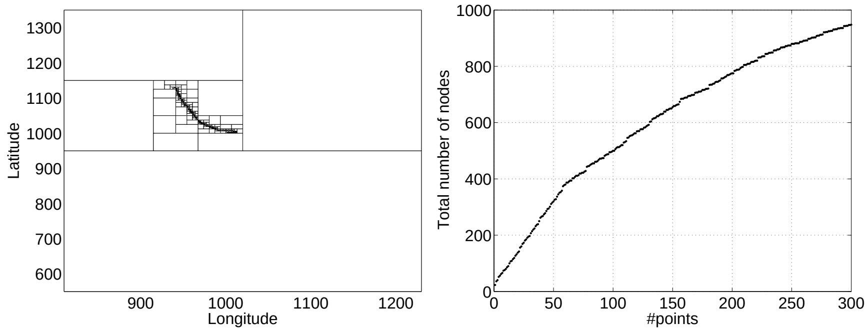
$$k \leq \min \left\{ ni^* = nd \log_2 \left(\frac{1}{h} \right)^d, 2m^d - 1 \right\} \quad .$$

In the analyses, we are working with data from actual flights that generally fly on established routes between established waypoints. Thus, we do not expect actual aircraft trajectory data to evenly fill the entire airspace. A dynamic SRP structure as discussed in Section 6.5 allows the SRP tree to grow and shrink as needed such that the number of nodes required stabilize as the number of aircraft in airspace reaches a steady state. Figure 6.12 shows, in log scale, the ratio of the number of nodes in an SRP to the number of cells in a regular grid given

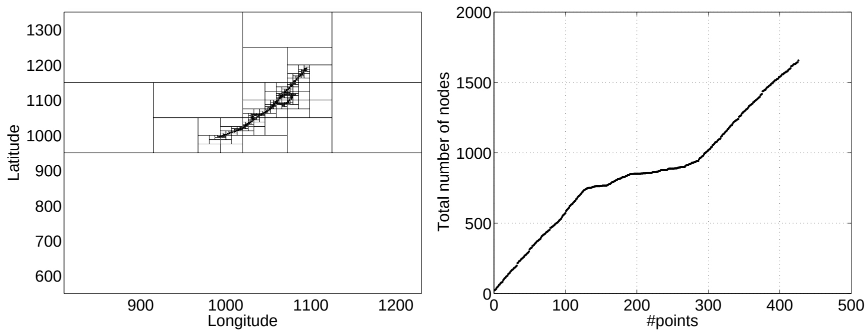
λ for time-ordered position data on a good and bad weather day. Figure 6.12(a) shows the ratio (about 0.08) for individual trajectories; Figure 6.12(b) gives the ratio (about 0.16) for aggregated trajectories on a good and bad weather day. Finally, Figure 6.12(c) gives the ratios at various time intervals of the dynamic airspace structure in a $n_{1:m}$-presensed setting with time intervals of 30 min.

6.6.2 Time Complexity

The regular grid, being a random access container, allows for instantaneous insertions and deletions in any dimension. For an SRP trajectory, if we measure time in units of nodes traversed or created, then each data point requires i^* time units to reach the box at depth i^* that encloses it. Thus, the time complexity is $\mathcal{O}(i^*) = \mathcal{O}(d \log_2 m)$, i.e., linear in d (when h is identical in each of the d coordinates). This is slower than a grid but, given that radar data arrives in bursts of 4-12 s, an SRP trajectory can easily be updated in time for online analysis. If the insertion time of a data point into a regular grid requires one CPU time unit, then we will need i^* CPU time units to insert a data point that is enclosed by a box at depth i^* of the SRP trajectory. Thus, for a given data point, the ratio of its insertion time into a regular grid to its insertion time into an SRP trajectory is i^*. Figure 6.13 gives the CPU times needed for such online analysis corresponding to the structures developed in Figure 6.10(d). We observe that the timings are reasonable to handle data arriving in bursts of 10 s from the radar. All of our programs were run on a machine with dual Intel X5670 2.93Ghz 6 core Xeon CPUs, 48GB of RAM, 2 x 320GB 15K serial attached small computer system interface (SAS) hard drives, and an OpenSuSE 11.2 (x86 64) operating system.

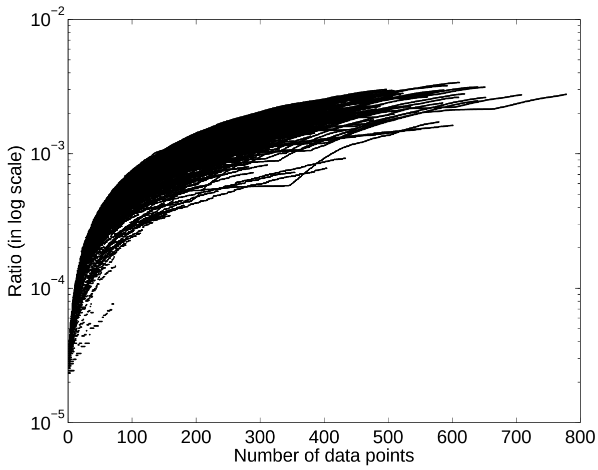


(a) Trajectory of aircraft A.

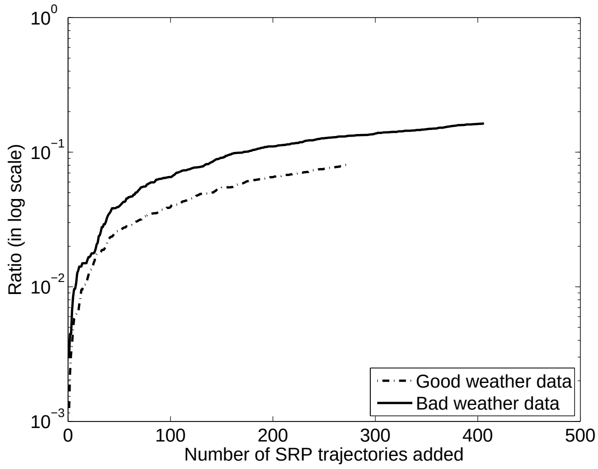


(b) Trajectory of aircraft B.

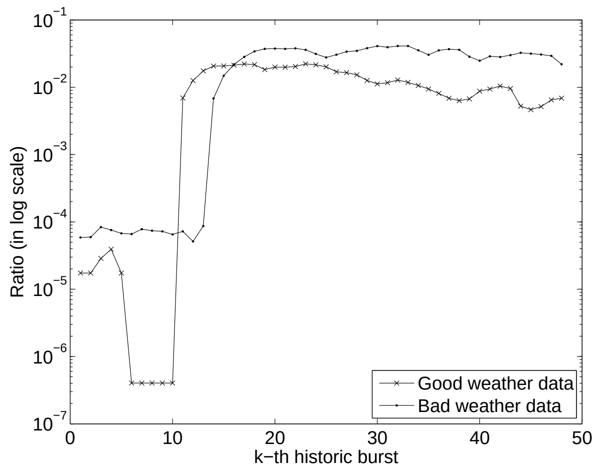
Figure 6.11: Total number of nodes needed to represent two aircraft trajectories.



(a) Individual SRP trajectories.



(b) Aggregate SRP trajectories.



(c) Dynamic airspace structure ($n_{1:m}$-presensed).

Figure 6.12: Ratio of the number of nodes in an SRP to the number of cells in a regular grid.

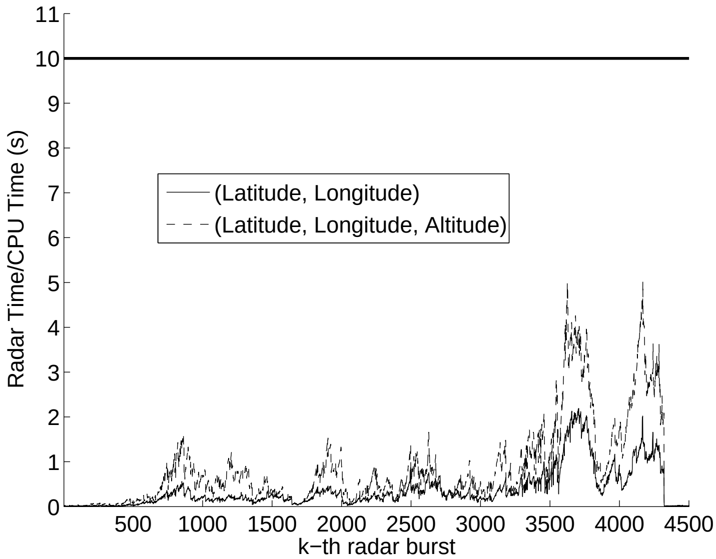


Figure 6.13: CPU times used when building dynamic structures for online analysis.

Chapter 7

Discussions and Conclusions

We introduced SRPs, an information structure extended from RPs, to handle massive data problems in the context of density estimation, and the analysis of spatio-temporal data in the context of aviation systems research. We note that bisections are only allowed along the midpoint of the first widest coordinate to produce an RP, which may not be appropriate for applications that would benefit from a more flexible partitioning scheme. However, such restrictive bisections allow arithmetic to be naturally extended to SRPs, which we have utilised as (i) an approach to tackle the smoothing problem in density estimation, and (ii) to aggregate/separate aircraft trajectory data accordingly for potential learning tasks.

We showed how an SRP histogram that is asymptotically consistent in the L_1-setting can be constructed using a randomised priority queue based on SEB, and showed that its space and time requirements in the post-constructive setting is no larger that $O(nd)$ on the condition that our memory-justified bound $\overline{m}_n$ satisfies Equation 3.2, which involved space requirements needed to store a box associated with a node of the SRP and the height value.

We avoid methods that involve parameter-tuning and exhaustive searches when approaching the smoothing issue which requires the selection of a suitable $\overline{k}_n$ to produce an optimal histogram estimate. Instead we proposed to obtain a posterior mean histogram estimate which can be obtained from the average of a number of independent random samples of partition states from the posterior probability distribution. Such random samples of states are taken from a Metropolis-Hastings Markov chain over a finite state space $\tilde{S}_n$ using an appropriate Catalan prior. Here we take full advantage of the ability to perform arithmetic over SRPs to obtain the average.

We showed that our MCMC sampler performs well for massive data problems in high dimensional settings using simulations from multivariate uniform densities and mixtures of uniform densities that approximate multivariate Gaussian and Rosenbrock densities. We chose mixtures of uniform densities to simplify the multivariate integrations in absolute error evaluations. Given that the space of regular paving histograms with their countably infinite family of partitions in $\mathbb{S}_{0:\infty}$ are dense in the space of continuous densities over the root box, one can come uniformly close to any continuous density when given enough data points from the space of SRP histograms. The rate of uniform convergence, in terms of the number of splits, to a particular density within a given class, from the space of SRP histograms for a given sample size n would shed light on the statistical efficiency of our estimator. However, we are able to achieve low integrated absolute errors because of the large sample sizes we assume in our massive data setting. Such a large number of data points allow our Markov chains to dive as deep as necessary into the SRP space of trees to get to the partitions of the true densities that are generating the data.

We are currently using heuristics for determining when the chain has converged. It is not straightforward to produce perfect samples from the posterior distribution on the state space of adaptive SRP histograms due to its size and combinatorially complicated transition structure. We used one prior distribution in this study and show that the simulations are reasonably good when the size of the data is large. The effect of other prior distributions on the posterior (especially for smaller sample sizes) and other base chains on mixing time should be further explored.

The methods developed in this project consider only partitions formed by successive bisections of the data space on the widest dimension of the box being bisected. There are two important constraints in this: first, we consider only bisections, or division of a box into two equal-volume halves. Secondly, the choice of basis, or dimension to split on, is not directly data driven. These constraints give the partitions very attractive qualities from the point of view of creating binary tree structures to represent the histograms, and also mean that the process of adding and averaging histograms is relatively straightforward.

A typical nested sequence of Pólya tree partitions is the sequence

$$\{s_0, s_{1\,1}, s_{2\,2\,2\,2}, \ldots\} = \{s_{i\,i\,\cdots i\,i}\}_{i=0}^{\infty}, \quad s_{i\,i\cdots i\,i} \in \mathbb{S}_{(2^1)^i-1}, \quad \boldsymbol{x}_\rho \in \mathbb{IR}^1 \ .$$

More generally, for any root box $\boldsymbol{x}_\rho \in \mathbb{IR}^d$, a typical nested sequence of Pólya tree partitions with the uniformly distributed base measure is the sequence

$$\{s_{di\,di\,\cdots di\,di}\}_{i=0}^{\infty}, \quad s_{di\,di\cdots di\,di} \in \mathbb{S}_{2^{di}-1}, \quad \boldsymbol{x}_\rho \in \mathbb{IR}^d \ .$$

Posterior inference of density estimates based on a Pólya trees (Lavine, 1992) for example can be strongly influenced by the choice of partition. This is a direct consequence of the fact that all random distributions from a Pólya tree have a common partition in a nested sequence of partitions. Extensions, such as a mixture of Pólya tree (Lavine, 1992) and randomized Pólya tree (Paddock et al., 2003), alleviate this problem by allowing the random distributions to have different partitions. Our space of SRP histograms $\tilde{\mathbb{S}}_n \subset \mathbb{S}_{0:\infty}$ is quite different from the partition generated by a nested sequence of Pólya trees because it contains about $\sum_{k=0}^{2^{di}-1} C_k$ many histograms with distinct partitions in $\mathbb{S}_{0:2^{di}-1}$ that are made of splits no larger than $2^{di}-1$ such that they are not all nested in the strict Pólya sense. Our density estimator is the posterior mean over this SRP histogram space, and therefore not reliant on a single partition of the sample space. Thus, our partitioning scheme and the space of histograms in $\mathbb{S}_{0:\infty}$, that are dense in the space of continuous densities on $\boldsymbol{x}_\rho$, seem to not influence the posterior estimates too much at least for the target densities in our simulation studies. The key motivation for our method is the ability to obtain density estimates for massive high-dimensional data sets. Our simple partitioning scheme of just splitting orthogonally along the midpoint of the first widest coordinate has been chosen precisely because it affords the computational efficiencies of histogram averaging that make our estimator computable for such massive high-dimensional data problems.

A current limitation of our posterior mean density estimate over SRP histograms is the approximation of the likelihood of the data given an s by the maximum likelihood value from the histogram on $\ell(s)$, the leaf boxes of s. A fully Bayesian approach involving a prior distribution on a class of simple functions over leaf boxes of s that are non-negative and integrate to 1 by an adaptation of Dirichlet distributions used in the constructions of classical Pólya trees [(Lavine, 1992); (Lavine, 1994)] would be a natural extension of our estimator. Such a Dirichlet process over SRP histograms would have the arithmetical efficiency of the dense space of SRP histograms as well as the fully Bayesian setting of classical Pólya trees and their mixtures. We hope that research in this integrative direction will continue.

We also proposed the use of SRPs for analysing spatio-temporal data using high frequency air position information. In particular, such data structures can be used to show which sections of airspace are most often occupied in different contexts (related to weather, time-of-day, etc.) by its unique rule that focuses computational resources on areas containing data points. The SRP has also been compared with a grid in terms of memory requirements and time complexity. It was shown that, although an SRP needs $\mathcal{O}(d \log_2 m)$ time units and is less efficient in terms of time compared to a grid, the SRP is more memory efficient when representing aircraft position data, especially in higher dimensions.

Thus, arithmetic operations (addition, subtraction, multiplication, etc.) were able to be developed for aircraft trajectory data: for example, aggregating trajectories over a given block of time, or taking the difference of two such aggregate trajectories on a good day and a bad day to see how much change in traffic there is between the two days. This is a step towards real-time classification and prediction of airspace usage, which in turn will be useful to monitor and manage air traffic controller workloads, local environmental impacts of aviation systems, airport and airspace throughput, etc.

It has also been shown that it is possible to reunite sibling nodes when information in the sibling nodes is not needed anymore. This helps prune the tree associated with the SRP trajectory and prevents the tree from over-growing (unnecessarily). With this capability, the SRP was transformed into a dynamic structure that allows tracking of flights in the airspace over time.

Finally, although SRPs have been used to represent the aircraft position in two or three coordinates through time, position, velocity, fuel-level and other real-valued measures of the aircraft could just have easily been used with a higher-dimensional root box. Arithmetic over SRPs generalize to any dimension and the memory requirements thus become significantly less demanding.

In closing, we have shown, through two applications, that though the space of SRPS are restrictive, but by making use of its restrictiveness and recursive nature, one can fully take advantage of its arithmetic extensions for desired purposes. We hope that future research will identify more applications with SRPs which are useful for handling massive data problems.

The algorithms described here are implemented in *MRS: a C++ class library for statistical set processing* and publicly available under the terms of the GNU General Public License from http://www.math.canterbury.ac.nz/~r.sainudiin/codes/mrs/.

Appendix A

The asymptotic partial sum of the Catalan numbers

Mattarei (2010) showed that the asymptotic partial sum of the Catalan numbers is as follows:

$$\sum_{k=0}^{n} C_k = \sum_{k=0}^{n} \frac{1}{k+1}\binom{2k}{k} \sim \frac{4^{(n+1)}}{3n\sqrt{\pi n}}$$

where "$\sim$" means that the ratio of the two sides tends to one as n tends to infinity.

The author proves the conjecture as follows:

By Stirling's approximation, $n! \approx n^n e^{-n}\sqrt{2\pi n}$. Thus $\binom{2n}{n} \sim \frac{4^n}{\sqrt{\pi n}}$. For positive sequences, if $a_n \sim b_n$ and $\sum_{k=1}^{n} b_k \to \infty$, then $\sum_{k=1}^{n} a_k \sim \sum_{k=1}^{n} b_k$. Now we have $\sum_{k=0}^{n} \frac{1}{k+1}\binom{2k}{k} \sim \sum_{k=0}^{n} \frac{1}{k+1}\frac{4^k}{\sqrt{\pi k}}$.

On one hand,

$$\sum_{k=0}^{n} \frac{1}{\sqrt{k}(k+1)}4^k \geq \frac{1}{\sqrt{n}(n+1)}\sum_{k=0}^{n} 4^k = \frac{1}{\sqrt{n}(n+1)}\frac{4^{n+1}-1}{3} \quad . \tag{A.1}$$

On the other hand, for any $1 \leq m \leq n$,

$$\sum_{k=0}^{n} \frac{1}{\sqrt{k}(k+1)}4^k \leq \frac{1}{\sqrt{m}(m+1)}\sum_{k=m}^{n} 4^k + \frac{1}{m+1}\sum_{k=0}^{m-1} 4^k \leq \frac{1}{\sqrt{m}(m+1)}\frac{4^{n+1}}{3} + \frac{4^m}{3(m+1)} \quad .$$

Take $m = \lfloor n - \log_4 n \rfloor$,

$$\sum_{k=0}^{n} \frac{1}{\sqrt{k}(k+1)}4^k \leq \frac{1}{3(n-\log_4 n+1)}\left(\frac{4^{n+1}}{\sqrt{n-\log_4 n}} + \frac{4^{n+1}}{n}\right) \quad . \tag{A.2}$$

Take Equation A.1 and A.2 together we get

$$\frac{1}{\sqrt{n}(n+1)}\frac{4^{n+1}-1}{3} \leq \sum_{k=0}^{n} \frac{1}{\sqrt{k}(k+1)}4^k \leq \frac{1}{3(n-\log_4 n+1)}\left(\frac{4^{n+1}}{\sqrt{n-\log_4 n}} + \frac{4^{n+1}}{n}\right) \quad .$$

Since $\sqrt{n} \sim \sqrt{n-\log_4 n}$ for large n, by the squeezing theorem, we get

$$\frac{1}{\sqrt{n}(n+1)}\frac{4^{n+1}-1}{3} \leq \sum_{k=0}^{n} \frac{1}{\sqrt{k}(k+1)}4^k \leq \frac{4^{n+1}}{3(n-\log_4 n+1)}\left(\frac{1}{\sqrt{n-\log_4 n}} + \frac{1}{n}\right) \tag{A.3}$$

$$\frac{1}{\sqrt{n}(n+1)}\frac{4^{n+1}}{3} \leq \sum_{k=0}^{n} \frac{1}{\sqrt{k}(k+1)}4^k \leq \frac{4^{n+1}}{3\sqrt{n}(n+1)}\left(1 + \frac{1}{\sqrt{n}}\right) \quad . \tag{A.4}$$

By multiplying $\sqrt{\pi}$ to the denominators in Equation A.4 we thus get

$$\sum_{k=0}^{n} C_k \sim \frac{4^{n+1}}{3(n+1)\sqrt{\pi n}} \ .$$

Bibliography

Agarwal, P. K., Gao, J., and Guibas, L. J. (2002), "Kinetic medians and kd-trees," in *Proceedings of the 10th Annual European Symposium on Algorithms*, Springer Verlag, pp. 5–16.

Akaike, H. (1974), "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, 19, 716–723.

Anderson, T. W. (1966), "Some nonparametric multivariate procedures based on statisticallly equivalent blocks," in *Multivariate Analysis*, ed. Krisnaiah, P. R., New York: Academic Press, pp. 5–27.

Baltrunas, L., Mazeika, A., and Bohlen, M. (2006), "Multi-dimensional Histograms with Tight Bounds for the Error," in *Database Engineering and Applications Symposium, 2006. IDEAS '06. 10th International*, pp. 105 –112.

Barron, A., Birgé, L., and Massart, P. (1999), "Risk bounds for model selection via penalization," *Probability Theory and Related Fields*, 113, 301–413.

Birgé, L. and Rozenholc, Y. (2006), "How many bins should be put in a regular histogram," *ESAIM: Probability and Statistics*, 10, 24–45.

Blanchard, G., Schäfer, C., Rozenholc, Y., and Müller, K.-R. (2007), "Optimal dyadic decision trees," *Mach. Learn.*, 66, 209–241.

Breiman, L., Friedman, J., Stone, C., and Olshen, R. (1984), *Classification and Regression Trees*, Chapman and Hall/CRC, 1st ed.

Callaham, M., DeArmon, J., A., C., Goodfriend, J., D., M.-M., and Solomos, J. (2001), "Assessing NAS Performance: Normalizing for the Effects of Weather," in *Proceedings of the 4th USA/Europe Air Traffic Management R&D Symposium*, Santa Fe, NM.

Castellan, G. (1999), "Modified Akaike's criterion for histogram density estimation," Technical report 99.61, Laboratoire de mathématiques, Université Paris XI, Orsay, France.

Celisse, A. and Robin, S. (2008), "Nonparametric density estimation by exact leave-p-out cross-validation," *Computational Statistics & Data Analysis*, 52, 2350–2368.

Chen, E. J. and Kelton, W. D. (2006), "Empirical evaluation of data-based density estimation," in *Proceedings of the Winter Simulation Conference*, Winter Simulation Conference, WSC '06, pp. 333–341.

Davies, P. L. and Kovac, A. (2004), "Densities, spectral densities and modality," *The Annals of Statistics*, 32, 1093–1136.

Devroye, L., Gyorfi, L., and Lugosi, G. (1996), *A probabilistic theory of pattern recognition (Stochastic Modelling and Applied Probability)*, New York: Springer.

Devroye, L. and Lugosi, G. (2001), *Combinatorial methods in density estimation*, New York: Springer.

— (2004), "Bin width selection in multivariate histograms by the combinatorial method," *TEST*, 13, 129–145, 10.1007/BF02603004.

Donoho, D. L. (2000), "High-Dimensional Data Analysis: The Curses and Blessings of Dimensionality," Aide-Memoire of a Lecture at AMS Conference on Math Challenges of the 21st Century.

Engel, J. (1997), "The multiresolution histogram," *Metrika*, 46, 41–57, 10.1007/BF02717165.

Fearnhead, P. and Prangle, D. (2011), "Constructing summary statistics for approximate Bayesian computation: semi-automatic approximate Bayesian computation," *Journal of the Royal Statistical Society Series B*, 74, 1–28.

Fisher, R. A. (1925), "Theory of Statistical Estimation," *Proceedings of Cambridge Philosophy Society*, 22, 700–725.

Friedman, N. and Getoor, L. (1999), "Efficient Learning using Constrained Sufficient Statistics," in *Proceedings of the 7th International Workshop on Artificial Intelligence and Statistic*, eds. Heckerman, D. and Whittaker, J., Morgan Kaufmann Publisher Inc.

Gelman, A. and Rubin, D. B. (1992), "Inference from Iterative Simulation Using Multiple Sequences," *Statistical Science*, 7, 457–472.

Gessaman, M. P. (1970), "A Consistent Nonparametric Multivariate Density Estimator Based on Statistically Equivalent Blocks," *The Annals of Mathematical Statistics*, 41, pp. 1344–1346.

Gibbons, P. B. and Matias, Y. (1999), "Synopsis Data Structures for Massive Data Sets." in *Symposium on Discrete Algorithms*, eds. Tarjan, R. E. and Warnow, T., ACM/SIAM, pp. 909–910.

Harlow, J., Sainudiin, R., and Tucker, W. (2012), "Mapped Regular Pavings," *Reliable Computing*, 16, 252–282.

Hartigan, J. A. (1996), "Bayesian histograms," in *Bayesians Statistics 5*, eds. Bernardo, J. M., Berger, J. O., Dawid, A. P., and Smith, A. F. M., Oxford, England: Claredon Press, pp. 211–222.

Hearne, L. and Wegman, E. (1994), "Fast Multidimensional Density Estimation based on Random-width Bins," Tech. rep., George Mason University, Fairfax, VA.

Hilbert, M. and López, P. (2011), "The World's Technological Capacity to Store, Communicate, and Compute Information," *Science*, 332, 60–65.

Hurter, C., Tissoires, B., and Conversy, S. (2009), "FromDaDy: Spreading Aircraft Trajectories Across Views to Support Iterative Queries," *IEEE Transactions On Visualization and Computer Graphics*, 15, 1017–1024.

Jaulin, L., Kieffer, M., Didrit, O., and Walter, E. (2001), *Applied Interval Analysis with Examples in Parameter and State Estimation, Robust Control and Robotics*, London: Springer-Verlag.

Kanazawa, Y. (1992), "An optimal variable cell histogram based on the sample spacings," *Annals of Statistics*, 20, 291–304.

Kettenring, J. R. (2009), "Massive Datasets," *Wiley Interdisciplinary Reviews: Computational Statistics*, 1, 25–32.

King, N. (2001), "Hay Stacks, Few Needles," Wall Street Journal, May 23, 2001.

Klemelä, J. (2007), "Density estimation with stagewise optimization of the empirical risk," *Machine Learning*, 67, 169–195.

— (2009), "Multivariate histograms with data-dependent partitions," *Statistica Sinica*, 19, 159–176.

Kogure, A. (1987), "Asymptotically optimal cells for a histogram," *Annals of Statistics*, 15, 1023–1030.

Komarek, P. and Moore, A. W. (2000), "A Dynamic Adaptation of AD-trees for Efficient Machine Learning on Large Data Sets," in *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 495–502.

Kontkanen, P. and Myllymki, P. (2007), "MDL histogram density estimation," in *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*.

Kuhn, K. (2008), "Analysis of Thunderstorm Effects on Aggregate Aircraft Trajectories," *Journal of Aerospace Computing, Information, and Communication*, 5, 108–119.

Lambert, D. (2003), "What Use is Statistics for Massive Data?" in *Leture Notes – Monograph Series, Crossing Boundaries: Statistical Essays in Honor of Jack Hall*, eds. Kolassa, J. E. and Oakes, D., Institute of Mathematical Sciences, pp. 217–228.

Lambert, D. and Pinheiro, J. C. (2001), "Mining a stream of transactions for customer patterns," in *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2001)*, ACM Press, pp. 26–29.

Lauritzen, S. L. (1983), "Notions of Sufficiency," in *Proceedings of the 44th session of the International Statistical Institute*, Madrid, Spain: International Statistical Institute.

Lavine, M. (1992), "Some aspects of Polya tree distributions for statistical modelling," *The Annals of Statistics*, 20, 1222–1235.

— (1994), "More aspects of Polya tree distributions for statistical modelling," *The Annals of Statistics*, 22, 1161–1176.

Lester, E. and Hansman, R. (2007), "Benefits and Incentives for ADS-B Equipage in the National Airspace System," Master's thesis, Massachusetts Institute of Technology, Cambridge.

Lugosi, G. and Nobel, A. (1996), "Consistency of Data-Driven Histogram Methods for Density Estimation and Classification," *The Annals of Statistics*, 24, 687–706.

Massart, P. (2007), *Concentration Inequalities and Model Selection*, vol. 1896, Berlin: Springer.

Mattarei, S. (2010), "Asymptotics of partial sums of central binomial coefficients and Catalan numbers," ArXiv.0906.4290v3.

McGarvey, G. and Cloitre, B. (2005), "Sequence A000108, The On-Line Encyclopedia of Integer Sequences," published electronically.

Moore, A. W. and Soon Lee, M. (1998), "Cached Sufficient Statistics for Efficient Machine Learning with Large Datasets," *Journal of Artificial Intelligence Research*, 8, 67–91.

Moore, R. E. (1967), *Interval analysis*, Englewood Cliffs, New Jersey: Prentice-Hall.

Neumaier, A. (1990), *Interval methods for systems of equations*, Cambridge: Cambridge University Press.

Paddock, S., Ruggeri, F., Lavine, M., and West, M. (2003), "Randomized Pólya tree models for nonparametric Bayesian inference," *Statistica Sinica*, 13, 443–460.

Rissanen, J., Speed, T., and Yu, B. (1992), "Density estimation by stochastic complexity," *IEEE Transactions on Information Theory*, 38, 315–323.

Rozenholc, Y., Mildenberger, T., and Gather, U. (2009), "Constructing Irregular Histograms by Penalized Likelihood," Pre-print submitted to Elsevier.

— (2010), "Combining regular and irregular histograms by penalized likelihood," *Computational Statistics & Data Analysis*, 54, 3313–3323.

Rudemo, M. (1982), "Empirical choice of histograms and kernel density estimators," *Scandinavian Journal of Statistics*, 9, 65–78.

Sainudiin, R. and Harlow, J. (2010), "A C++ Class Library for Statistical Set Processing," in *Short Communication in Mathematical Software*, International Congress of Mathematicians, Hyderabad, India, p. 670.

Sainudiin, R. and York, T. (2005), "An Auto-validating Rejection Sampler," Technical report bu-1661-m, BSCB Dept., Cornell University, Ithaca, New York.

Samet, H. (1990), *The design and analysis of spatial data structures*, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Scott, D. and Sain, S. (2005), "Multidimensional Density Estimation," *Handbook of Statistics*, 24, 229–261.

Scott, D. W. (1985), "Averaged Shifted Histograms: Effective Nonparametric Density Estimators in Several Dimensions," *The Annals of Statistics*, 13, 1024–1040.

Stanley, R. P. (1999), *Enumerative Combinatorics, Vol. 2*, Cambridge: Cambridge University Press.

Stone, C. (1982), "Optimal Global Rates of Convergence for Nonparametric Regression," *The Annals of Statistics*, 10, 1040–1053.

Tao, Y. and Papadias, D. (2005), "Historical spatio-temporal aggregation," *ACM Trans. Inf. Syst*, 23, 61–102.

Taylor, C. C. (1987), "Akaike's Information Criterion and the Histogram," *Biometrika*, 74, pp. 636–639.

Teng, G., Kuhn, K., and Sainudiin, R. (2012), "Statistical regular pavings to analyze massive data of aircraft trajectories," *Journal of Aerospace Computing, Information and Communication*, 9, 14–25.

Wand, M. (1997), "Data-based choice of histogram bin width," *The American Statistician*, 51, 59–64.

Wedner, K. and Hofschuster, W. (2001), "C-XSC 2.0 - A C++ Class Library for Extended Scientific Computing," Tech. rep., Universitat Wuppertal.

Welford, B. P. (1962), "Note on a method for calculating corrected sums of squares and products," *Technometrics*, 4, 419–420.

Wilkerson, J. T., Jacobson, M. Z., Malwitz, A., Balasubramanian, S., Wayson, R., Fleming, G., Naiman, A. D., and Lele, S. K. (2010), "Analysis of emission data from global commercial aviation: 2004 and 2006," *Atmospheric Chemistry and Physics*, 10, 6391–6408.